AFRL-RH-WP-TR-2010-0120

3D Visualizations of Abstract DataSets

Elisabeth Fitzhugh
Eric Smith
SRA International, Inc.

Sharon Ellis
Battlespace Visualization Branch
Warfighter Interface Division

August 2010
Final Report

Approved for public release; distribution is unlimited.

See additional restrictions described on inside pages

AIR FORCE RESEARCH LABORATORY
711 HUMAN PERFORMANCE WING,
HUMAN EFFECTIVENESS DIRECTORATE,
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433
AIR FORCE MATERIEL COMMAND
UNITED STATES AIR FORCE

NOTICE AND SIGNATURE PAGE

AFRL-RH-WP-TR-2010-0120 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

//signed//
Sharon A. Ellis
Program Manager
Battlespace Visualization Branch

//signed//
Jeffrey L. Craig
Chief, Battlespace Visualization Branch
Warfighter Interface Division

//signed//
Michael A. Stropki
Chief, Warfighter Interface Division
Human Effectiveness Directorate
711 Human Performance Wing

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
30-07-2010	Final	July 2009 to July 2010

4. TITLE and SUBTITLE
3D Visualizations of Abstract Datasets

5a. CONTRACT NUMBER
FA8650-09-D-6939 TO 11

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
62202F

6. AUTHOR(S)
Elisabeth Fitzhugh*
Eric Smith*
Sharon Ellis**

5d. PROJECT NUMBER
7184

5e. TASK NUMBER
11

5f. WORK UNIT NUMBER
71841145

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
SRA International, Inc.*
5000 Springfield Street
Suite 200
Dayton OH 45431

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Materiel Command**
Air Force Research Laboratory
711 Human Performance Wing
Human Effectiveness Directorate
Warfighter Interface Division
Battlespace Visualization Branch
Wright-Patterson AFB OH 45433

10. SPONSOR/MONITOR'S ACRONYM(S)
711 HPW/RHCV

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-RH-WP-TR-2010-0120

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
88 ABW Cleared 10/03/2010; 88ABW-2010-5330.

14. ABSTRACT The 3D Visualizations of Abstract Datasets effort is part of a series of 3D experimentation with dual goals of exploring the applicability of 3D environments to abstract network presentations and defining guidance for optimal display design. The series encompasses a survey of 3D military applications, a subjective metrics study of user preferences for 3D display types, and two sets of experimentation documenting user performance interpreting node depth in 2.5D and 3D abstract network displays. The current study (not yet completed) focuses on user ability to interpret node depth, under two levels of network complexity, employing several types of depth interpretation aids. The experimental hypothesis anticipates diminishing returns in speed and accuracy as interpretative aids increase scene clutter. The first aiding condition provides lines of perspective through an artificial floor and contrasts labeled tick marks, lanes, and grid lines. The second contrasts no shadows, drop shadows and drop lines.

15. SUBJECT TERMS
3D displays, 2.5D displays, abstract network visualizations, depth perception, human performance

16. SECURITY CLASSIFICATION OF: a. REPORT	b. ABSTRACT	c. THIS PAGE	17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON / 19b. TELEPONE NUMBER *(Include area code)*
UNCLASS	UNCLASS	UNCLASS	SAR	84	Sharon Ellis / (937) 255-1217

THIS PAGE IS INTENTIONALLY LEFT BLANK

Table of Contents

List of Figures

List of Tables

Executive Summary

Command interest in the applicability of 3D technologies as a visualization solution prompted an exploratory effort to identify and respond to gaps in the understanding of how to effectively employ 3D technologies in military applications. The 3D Visualizations of Abstract Datasets effort is part of a series of 3D experimentation with dual goals of exploring the applicability of 3D environments to abstract network presentations and defining guidance for optimal display design. The series encompasses a survey of 3D utilization in military applications, a subjective metrics study documenting user preferences for 3D display types, and two sets of experimentation documenting user performance interpreting node depth in 2.5D and 3D abstract network displays.

The current study (not yet completed) focuses on user ability to interpret node depth, under two levels of network complexity, employing several types of depth interpretation aids. The purpose of the study is to determine appropriate levels of depth interpretation aiding for abstract displays such as may be used to represent extensive social networks (e.g., multinational terrorism individual and cooperative organizations). The need for experimentation was suggested by the mixed performance results obtained by experimenters investigating depth and altitude perception in airspace management and airspace route planning—simulated reality visualizations that employ altitude and heading as well as terrain information. Similar testing was not found for abstract network representations, which would provide fewer in-scene location and navigation memory cues.

Mechanisms exist to enhance depth interpretation; these include lines of perspective, atmospheric effects, size, and the insertion of shadows and drop lines to visually connect nodes to an artificial ground. The experimental hypothesis anticipates diminishing returns in speed and accuracy as interpretative aids increase scene clutter. The first aiding condition provides lines of perspective through an artificial floor and contrasts labeled tick marks, lanes, and grid lines. The second contrasts no shadows, drop shadows and drop lines. The effects of target node location on both y and z axes in depth interpretation performance are of interest. This report describes experimental design; experimental results will be reported on separately.

1.0 Introduction

This report describes the experimental plan developed for Information Operations/Cyber Exploitation Research (ICER) Task 11, 3D Visualizations of Abstract Datasets. The experimental plan and subsequent experimentation was directed by the Air Force Research Laboratory's (AFRL's) Human Effectiveness Directorate, Warfighter Interface Division, Battlespace Visualization Branch (711 HPW/RHCVZ). The objective of Task Order 11 is to investigate the saliency of real-world depth cues for 2D, perspective (2.5D), and "true" (3D) visualizations of abstract datasets through basic research. The research plan is envisioned to determine whether depth cues employed by display designers for depicting real-world scenes on a flat surface can be applied to create a perception of depth for abstract visualizations. The task focuses on determining which application of depth cues, singly or in combination, effectively enhanced the perception of depth in abstract scenes within 2.5D and 3D mediums for effectiveness of information portrayal. At the time of this writing, experimentation is still in progress; therefore, experimental analysis will be completed after the submission of this report. This report focuses on the need for experimentation and details the experimental design. Results will be discussed in a separate document, to be published at a later date.

Air Force interest in 3D visualization solutions spans 3D avatars for new recruits (Hand, 2010), virtual reality visualizations of both aerospace and earthbound operational environments, 3D medical imaging, 3D training environments, and 3D visualizations for knowledge management. However, 711 HPW/RHCVZ's initial investigation determined that, (barring such notable exceptions as SPAWAR-sponsored research on operational use of 3D (e.g., St John & Cowen, 1998; St John, Smallman, & Cowen, 2005; Smallman & St John, 2005; Smallman, Cook, Manes, & Cowen, 2007) and operationally relevant work done at the University of Illinois (e.g., Wickens & Carswell, 1995; Wickens & May, 1995; and Wickens, 2000), there were few published guidelines for scene design and little research on the efficacy of 3D visualizations in operational applications. However, several reports suggested that actual 3D efficacy might not match the level of utility anticipated by 3D enthusiasts (Andre & Wickens, 1995; Wickens, 2000; St John, Smallman & Cowen, 2000). To add to the body of knowledge, RHCVZ conceived a series of exploratory efforts with the dual goals of 1) examining the effectiveness of 3D as a display medium for complex, abstract, network displays and 2) developing design guidance for configuring network scene characteristics, including use and configuration of depth cues and the use of depth cue interpretive aids (e.g., drop shadows and drop lines). The current experiment is the second in the series. A separate but related experimental effort was part of a 3-pronged series of studies to create a database of metrics ("a 3D style guide") to assess 3D displays for best, most cost-effective military applications. It investigated subjective metrics for a range of 3D display types. That effort will be documented separately.

1.1 Background

The dual questions of the applicability of 3D and the appropriate use of 3D in visualization development are far from trivial. The military community shares commonly

held public requirements for advanced visualization for complex multidimensional weather and environmental data sets. General John Jumper suggested exploitation of 3D mapping capabilities to provide nth degree strike precision—he envisioned layers of xyz coordinates (x′y′z′) for different floors within a target building (Jumper, 2003). Airspace management is an endeavor, shared by public and military sectors, that appears to lend itself to judicious application of 3D technologies. Advanced visualizations to manage complex communications networks (including integrated space-based assets) are sought across civilian-military boundaries; 3D visualizations have been proposed to delineate large-scale communications enterprises. 3D applications have been suggested for management and exploitation of the wealth of documents published on or available through the internet, which links public and limited access repositories (e.g., libraries, agency databases, etc.). For many years, the intelligence community (military and civilian) has sought means to manage and exploit its access to intelligence information that exceeds the community's ability to parse and process it. This is a mission-critical data mining application for visualization of and navigation through large quantities of disparate types of information (reports, message traffic, multimedia files, sensor system feeds) of varying quality (in terms of confidence levels and freshness) from disparate sources. Another application for complex multidimensional data visualization is presented by the large-scale networks that can be built to depict the intricate social networks of globally connected terrorist groups. Of course, 3D applications also exist and have been employed for route and topographic mapping, simulation-based mission rehearsal, and simulation-based training.

3 D research issues needing to be addressed include the use for groups (e.g., difficulties providing equivalent views to differently positioned viewers), the use with augmenting equipment (e.g., the need for special glasses), and the potential for visual fatigue and motion sickness. These issues are common to both realistic and abstract 3D applications. The abstract applications raise other flags as well. Do people really locate and retrieve information better in 3D presentations? Do they comprehend multidimensional data better presented in 3D? Does the artificially generated 3D provide affordances that are missing from the true 3D experience, or instead, does one experience the same visual impediments (e.g., occlusion) with less (or more awkward) freedom of movement? As the body of research begun by SPAWAR indicates, in some cases 3D is not superior to 2D. For instance, interpretation of altitude and heading for multiple objects, when viewed from a single perspective, can be visually challenging. Occlusion definitely can be a factor as well. When we enthuse about the realism 3D provides, we simply fail to recall how visually confusing some of our interactions in the real world can be—and how often it is necessary to change position or to enter further into a scene in order to fully absorb it.

These and other questions about the efficacy of 3D in abstract applications are still open to research. However, another set of issues, at an immediate level, face the designer who attempts to use the 2.5D and 3D programs that are currently available. There are multiple important decision points in setting up a scene. These include, but are not limited to positioning of the lighting, intensity and color settings for both atmospheric effects and shadowing, object color selection, saturation, and luminosity, etc. If one were building a

maximally realistic scene, one would choose settings that provide a maximally realistic experience. But is that really the goal in abstract scenes? Within the natural world, depth cues, such as atmospheric effects and shadows, aid in depth interpretation but also obscure scene information. What are the optimal settings for interpretability of abstract data sets presented as networks or geometric shapes? The guidance for designers is not clear. In response to these open questions, AFRL has begun a series of explorations into the requirements of abstract data visualizations. The investigations intend to distinguish the visually effective applications for 2.5D and 3D technologies from those that, while they may be visually attractive, are of limited utility. A second intent is to obtain a clearer understanding of how interpretive cues may best support comprehension of complex multidimensional data sets in abstract presentations. Empirical evidence obtained from the exploratory efforts will provide design guidance to optimize data visualization.

3D technologies come in several forms. Stereoscopic 3D, perhaps the most familiar of the 3D techniques, projects two slightly offset views of the same image in an attempt to mimic the external feeds to the human binocular vision system. 3D perspective displays, also called 2.5D displays, employ software techniques to create an oblique (typically, a 30° or 45° angle) view of space that employs visual depth cues to provide the appearance of a 3D environment on a flat screen. 3D immersive techniques permit the user to physically enter the projected scene to interact within the 3D space. The experiments reported herein focused on 2.5D displays, as their relatively low costs and lack of special hardware requirements recommend them for the most immediate use in military operational settings.

The long-term goal of the research effort is twofold: a) to determine whether, when, and what type of 3D visualizations are preferable to 2D visualizations and b) to provide guidance for development of maximally usable and useful information displays. The effort is anticipated to include a series of studies using different display media, under varying conditions, in a range of applications. The initial pilot study, conducted in the 2008-2009 timeframe, employed 2.5D display techniques and was focused on the collection of basic performance data to support display design decisions for the most readily available and affordable type of simulated 3D experience in a frequently proposed use—network visualization (Fitzhugh et al., 2009). The experiment manipulated size and atmospheric effects (fog color and fog intensity) depth cues in linear and abstract network node displays in order to investigate their relative importance in depth interpretation.

The first question addressed relative size—was its strength as a depth cue so compelling as to preclude its being used as a coding dimension (e.g., size = importance, size = number of links, size = suspense date). In fact, its removal as a cue made depth interpretation much more difficult and much less accurate. Further, relative size is so closely associated with depth that to use it for an alternative meaning is to risk misinterpretation. The second question addressed the use of fog as a depth cue, alone and in conjunction with relative size. Fog enhanced both conditions, but its effect in the presence of relative size, although positive, was negligible. As the sole depth cue it provided significant assistance; its absence impeded both accuracy and response time.

The third question addressed whether the fog color extremes, black and white, were equivalent cues. The response was not clear. In the constant terminal fog condition, white fog outperformed black; in the adjusted (more rapidly intensified) terminal fog condition, black fog outperformed white. It was suggested that more experimentation is needed to derive guidelines for fog color; however, as fog options within 3D software span the spectrum of color, further experimentation would have been recommended in any case. The final question addressed whether positions at the outer edge of the foveal field of vision would be interpreted less well than those more centrally located. To this end, a linear 12-node array and a 12-node scattered array were employed. In pre-testing there were strong effects from position, but during experimentation, array alone did not yield significant differences and position appeared to have no real bearing on performance. The shift to use of a larger monitor and different viewing distance may have inadvertently changed the limits for the foveal field of view. It was suggested that this issue should be also be revisited in future experimentation.

The current experiment is a continuation of the investigation into the use of depth cues. The importance of the appropriate design use of depth cues cannot be overstated. Almost all of our understanding of the use of real-world cues to convey a feeling of depth on a 2D surface comes from the world of art. Bardel (2001) discusses the use of a variety of depth cues including size and texture gradients, linear perspective, occlusion and shadowing, focus, atmospheric effects and relative intensity for creating a perception of depth on a flat surface. Weiskopf and Ertl (2002) discuss color-oriented depth cuing based on linear transformations in tristimulus space resulting in intensity cuing, saturation cuing, and hue gradients, as well as, combinations of saturation and intensity for effective depth applications. Boyd (2000) investigated the effects of shape-from-shading and motion parallax. Their integrated results suggest that the removal of cues such as luminance and speed leaves unpracticed subjects virtually unable to acquire depth information given only a single 3D cue to supplement 2D cues. This illustrates the difficulties that are inherent understanding and translating how depth is experienced from the visual system to flat surface mediums.

Current state-of-the-art 3D visualization software packages used by display artists apply user-defined methods to display linear perspective, shadowing and size as depth cues to create realistic depictions of environments of interest. The software also allows the user to manually select and move lighting sources and to apply various applications of the atmospheric cue fog, to create the illusion of depth in a static display. However, ability to control scene characteristics is seldom supported by detailed guidance in how to determine optimal settings for different situations (Fitzhugh, Dixon, Aleva, Smith, Ghrayeb & Douglas, 2009). How the user should manipulate the lighting, as well as the depth cues (e.g., fog, linear perspective, etc.), for optimum visual results is not explained. A recent literature survey found few established guidelines for use of the various settings available with these software packages to create 3D environments (Dixon, Fitzhugh, & Aleva, 2009). While few experiments have been conducted to determine which depth cues are most applicable to abstract environments or what combinations work well for design of abstract visualizations, recent, limited laboratory experimentation has shown that 3D visualization software depth cues can create confusing effects when applied to

abstract visualizations (Fitzhugh, et. al. 2009). Additionally, the depiction of certain depth cues and interpretive aids, (e.g., drop lines, drop shadows) may be experienced by the user as visual clutter (e.g., Aragon, 2004; Bemis, Schiller. & Ko, 2001; Liao. & Johnson, 2001; Smallman, Schiller, & Cowen, 2001; Smallman, Schiller, & Mitchell, 1999; Ware, 2003; Wickens, 2000; Wickens & May, 1995). It is important to determine at what point visual aids provide diminishing returns, obscuring rather than clarifying scene information.

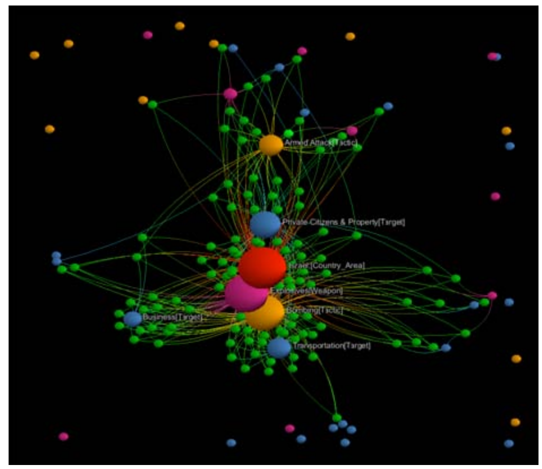
The phenomenal advances in computer processing, providing increasingly affordable 2.5D perspective renderings and 3D display technology, have sparked interest at the command level in the potential for military application across air, space, and cyber domains. Real-world depth cues have been used extensively in 3D gaming and virtual reality environments to create powerful experiences of environmental realism. Successful demonstrations of 3D technologies in multiple simulation applications as well as in battlespace environment depiction suggest potential employment of 2.5D and 3D technologies to convey complex, abstract, multidimensional information, in readily digestible forms, to facilitate quick and accurate identification of key attributes. The amount of data generated within aerospace and cyber domains is massive (in excess of one million data points) and rapid, effective interpretation of the data is critical to mission performance. The design of visualizations for these domains is especially challenging due to the quantity of information displayed (spatial data in millions of voxels, temporal data that may exceed thousands of time steps, and variables that may be in the hundreds), the disparate goals and needs of its multiple users, and the requirement to maintain situation awareness. In order to effectively visualize the deluge of data, guidance for the most effective application of 3D visual cues must be established. In addition, guidelines are needed for the determination of the best, most cost effective display medium to employ.

Much work has been done on interpretation of and navigation through realistic 2.5D and 3D environments but little to no information is available referring specifically to abstract environments- defining specific requirements for these environments will enhance design effectiveness (Dixon et.al, 2009 and Fitzhugh et. al., 2009). Among the visual cues in 3D simulation, depth cues are highly salient (Naikar, 1998). The study described in this report is one of an ongoing series of experiments conceived by 711 HPW/RHCVZ to address design decisions applicable to potential military use of 3D visualization capabilities; it and the pilot study that preceded it (Fitzhugh, et al, 2009) explore the use of depth cues in scene interpretation.

Figure 1 presents three distinct uses of 3D to visualize operational information. The top illustration shows a 3rd person view of an A-10 entering airspace over a realistic ground scene. The bottom left is a screenshot of how 3D might be used to visualize air traffic management approaching an airport. The bottom right depicts filtered, weighted terrorism data using the University of California-Davis VIDI Research Group’s OntoVis system; it is focused on terror attacks within Israel and uses color coding to identify node types (green for attacks, purple for weapons employment, brown for attack tactics and blue for attack targets). Size indicates numbers of instances. Line labels are not shown.

MetaVR VRSG screen capture of a 3D model A-10 entity approaching a high-resolution modeled Afghan village on MetaVR's Afghanistan virtual terrain.

Example screen from Barco Airways Collaborative Arrival Management (CAM) software.



Terrorism data set to highlight attacks at Israel. Node types are color-coded; size shows degree.

Figure 1. Comparison of uses of 3D to support operational environment understanding/awareness

These three examples illustrate how differently one may employ 3D visualization capabilities in operational applications, but they also illustrate how differently one might approach optimal use of 3D technology in terms of visualization design and software settings and how differently one may wish to approach in-scene navigation. Manipulation of and navigation through the abstract network display, floating in an apparent vacuum, provides a distinctive set of challenges not shared by the visualizations that are visually connected to orientation-supportive and memory-jogging scene details.

The information collected in this experiment will contribute to a body of basic human performance guidelines as well as objective and subjective metrics for designing useful, usable 2.5D and 3D display visualizations and determining the best most cost effective application of the technology—including graphic displays for network operations (Havig et. al. 2008). The findings of this study will establish guidance for optimal depth cue application in the best, most cost effective display medium for effective visualization of abstract data sets across Air, Space and Cyber domains. Improving design choices and reducing design time represents not only a potential cost savings, but also improved overall operator situation awareness through enhanced ability to interact with and manipulate information critical for their mission. Specifically, the dual products of this research are anticipated to be a comparison of subject experience in 2.5D vs. 3D as well as the establishment of empirically derived guidance for depth cueing in 2.5D and 3D environments for the visualization (static and dynamic) of multi-dimensional abstract

datasets. At the time of this writing, the experiment being described is still in progress. As previously noted, this report provides the experimental plan; experimental results will be reported on in a future effort.

2.0 Methods, Assumptions, and Procedures

2.1 Prework

The pilot study that preceded this effort included an extensive literature search encompassing both potential military applications of 3D technologies and visual cues employed for 3D scene interpretation (Dixon et al., 2009). During that literature review, the salience of depth cues and the importance of developing design guidance for depth cue instantiation were established. The need for further definition of the use of points of view (e.g., first person vs. third person, tethered vs. untethered, etc.) was also noted. Prior to the development of the current effort's experimental design, two new explorations were conducted within the body of 3D literature. The first sought guidance to determine appropriate commercial-off-the-shelf (COTS) and government-off-the-shelf (GOTS) software environments for use in experimentation. The second sought mathematical formulae for calculation of z-axis object positioning.

2.1.1 3D Environment Review

The review of 2.5D and 3D environment software spanned freeware, fee ware, and government-developed applications. The criteria employed for selection determination included flexibility of use and cost. The applications reviewed included, *3ds Max 2010*, *Maya 2009*, *Blender*, *Torque 3D Engine*, *Eye–Sys*, and *VOEUR*. One of the primary requirements was the ability to permit free fly-through navigation—necessary for any exploration of navigation abilities. That requirement reduced the field considerably. The assessment results are shown in Table 1.

Table 1. Comparison of 2.5D/3D Engines for 3D Environment Rendering and Manipulation

Feature/ Product	**3ds Max 2010**	**Maya 2009**	**Blender**	**Eye-Sys**	**Torque 3D Engine**	**VOEUR**
Type	**3D Modeling & Animation**	**3D Modeling & Animation**	**3D Modeling & Animation**	**3D Object/Attribute-based Bldg Tool**	**3D Gaming Engine with Dev Tools**	**Application Using JView 1.5 but can use 1.6**
Source/Cost						
Publisher	Autodesk	Autodesk	Blender Foundation	Interactive Data Visualization	GarageGames	SRA and AFRL
COTS	Yes	Yes	No	Yes	YES	NO
Cost	$3,500	$2000 Complete; $5000 Ultimate	Free	SDK, $2000; Single Seat $4000	Basic $250, Professional $1000, Studio $3000	Kind of Free
Overall UI (ease of use)	5 out of 10	4 out of 10	3 out of 10	6 out of 10	Unknown for included tools. Full control for created app	Full control
Camera UI	9 out of 10	5 out of 10	6 out of 10	9 out of 10	Unknown for included tools. Full control for created app	Full control
Object UI	10 out of 10	8 out of 10	7 out of 10	9 out of 10	Unknown for included tools. Full control for created app	Full control
File Creation						
Create Image Files	Yes	Yes	Yes	Yes	I think only through screen captures	Yes
Create Movie Files	Yes	Yes	Yes	Yes	I don't think out of the box	I think so
View Capabilities						
Orthogonal View	Yes	Yes	Yes	Yes	Not sure	Yes, but might not be implemented the best in JView
1st Person Perspective	Yes	Yes	Yes	Yes	Yes	Yes
3rd Person Perspective	Yes, but only in final renders	Yes, but only in final renders	Yes, but only in final renders	Yes, but only in final renders, and this could be difficult	Yes	Yes, but would need some extra work to do
Intra-object Views	Yes for views	Yes	Yes	Yes	Yes	Yes
Free Fly-through Movement	No	No	No	Maybe with the SDK, but likely difficult	Yes	Yes
Fly-over View (Overview)	Yes, in movies	Yes, in movies	Yes, in movies	Yes	Yes, but it might be a little tricky, coding-wise	Yes
Snapshot Capability	No	No	No	No	Yes	Yes

Feature/ Product	3ds Max 2010	Maya 2009	Blender	Eye-Sys	Torque 3D Engine	VOEUR
Type	**3D Modeling & Animation**	**3D Modeling & Animation**	**3D Modeling & Animation**	**3D Object/Attribute-based Bldg Tool**	**3D Gaming Engine with Dev Tools**	**Application Using JView 1.5 but can use 1.6**
Position Controls	Yes	Yes	Yes	Yes	Yes in tools, and Yes in created app	**Yes**
Position With Mouse	Yes	Yes	Yes	Yes	Yes in tools, and Yes in created app	Yes
Position With Textfield	Yes	Yes	Yes	Yes	Yes in tools, and no in created app	Yes
Position With Script	Yes	Yes	I think so	Yes	Probably in both tools and created app	Kind of, but it would be a self made script most likely
Depth Cue Control						
Linear Perspective	Yes	Yes	Yes	Yes	Yes	Yes
Scale objects	Yes	Yes	Yes	Yes	Yes in tools, and Yes in created app	Yes
Materials Color	Full RGB	Full RGB	Full RGB	Full RGB	Full RGB	Full RGB
Supports Texture gradients	Yes	Yes	Yes	Yes	Yes	Yes
Shading in Real Time	Yes	Yes	Yes	Yes	Yes	Yes
Shading in Final Render	Yes	Yes	Yes	Yes	NA	NA
Shadows in Real Time	Yes	Yes	I don't think so	Yes	Yes	Yes, but it would take some research to find out how
Shadows in Final Render	Yes	Yes	Yes	Yes	NA	NA
Occlusion of objects	Yes	Yes	Yes	Yes	Yes	Yes
Fog Support	Volumetric Fog	Yes in both, but more in Ultimate	With use of particles	Yes (under State Manager)	Most likely	Yes using OpenGL
VOEUR-style Fog	I don't think so	I don't think so	No	Yes	Not sure	Yes
Materials (Attribute) Control	Yes	Yes	Yes	Yes	Yes	Yes
Materials Ease of Use	7 out of 10	5 out of 10	4 out of 10	8 out of 10	6 out of 10	2 out of 10
Material Texture maps	Yes	Yes	Yes	Yes	Yes	Yes
Material Bump maps	Yes	Yes	Yes	No	Probably, but not positive	I don't know for sure

Feature/ Product	**3ds Max 2010**	**Maya 2009**	**Blender**	**Eye-Sys**	**Torque 3D Engine**	**VOEUR**
Type	**3D Modeling & Animation**	**3D Modeling & Animation**	**3D Modeling & Animation**	**3D Object/Attribute-based Bldg Tool**	**3D Gaming Engine with Dev Tools**	**Application Using JView 1.5 but can use 1.6**
Material Ambient Color	Yes	Yes	No	Yes	Not sure	Yes
Material Diffuse Color	Yes	Yes	Yes	Yes	Yes	Yes
Material Specular Color	Yes	Yes for some material types	Yes	Yes	Not sure	Yes
Specular Highlight Color	Yes	No	No	Yes	Not sure	Not sure
Lighting Control	Yes	Yes	Yes	Yes	Yes	Yes
Lighting Ease of Use	6 out of 10	7 out of 10	6 out of 10	7 out of 10	Unknown	3 out of 10
Lighting Types	Point, Spot, Sun	Point, Spot, Sun, and more	Point, Spot, Sun and more	Point, Directional (Sun), Spot	Unknown, but probably the basic 3 plus more in Pro and up	Point, Directional (Sun), Spot
Rendering Capabilities						
Real time rendering	8 out of 10	8 out of 10	5 out of 10	8 out of 10	9 out of 10	7 out of 10
Final rendering	9 out of 10	9 out of 10	9 out of 10 but probably a lot slower	9 out of 10	NA	NA
Other						
Scripts support	MaxScript	MEL and Python	Python	Javascript	TorqueScript	Currently only hand-made, but could include JS or Python
Stereoscopic 3D	No	Yes in both Complete and Ultimate	Don't think so	No	Don't think so	I think there is enough support in JView for the Visbox type
Data Ingestion	Via Scripts & Plug-ins	Via Scripts & Plug-ins	Via Scripts & Plug-ins	COM Interface (req SDK), Shapefile, CSV, Excel, Access, txt, UCINET DL Files	I don't think out of the box, but it can always be added	I don't think out of the box, but it can always be added
Geospatial Awareness	No	No	No	Yes	No	Yes, but in JView World only
Import 3D file formats	Yes	Yes	Yes	Yes	Yes	Yes
					http://www.garagegames.com/products/torque-3d	
Definition of Terms:						

Feature/ Product	**3ds Max 2010**	**Maya 2009**	**Blender**	**Eye-Sys**	**Torque 3D Engine**	**VOEUR**
Type	**3D Modeling & Animation**	**3D Modeling & Animation**	**3D Modeling & Animation**	**3D Object/Attribute-based Bldg Tool**	**3D Gaming Engine with Dev Tools**	**Application Using JView 1.5 but can use 1.6**
Linear Perspective	Parallel lines in the scene appear to approach each other as they approach the apparent horizon					
Orthogonal View	Parallel lines in the scene always look parallel in a view					
Texture Gradients	Pattern details will look smaller and fainter with distance					
Position With *	Can move, rotate and resize the position of an object by using *					
Materials	Provide an object with properties that react to distinctively to light (e.g., rubber reflects light differently than steel)					
Texture maps	Maps an image to an object (e.g., a soup label on a cylinder)					
Bump maps	Maps a texture to an object (e.g., provides dimensionality to the finish on an object)					
Real time rendering	What the application will show when worked with it in real time					
Final rendering	What the application can make an image look like when allowed to "take its time" to create the image. Final rendering normally takes much longer than real time rendering, but also produce a superior image					
VOEUR-style Fog	This is the type of fog used in the first 3D experiment where fog has a start and stop point along with a color. Also called Terminal Fog					
	Every object that falls within the start and stop range of the fog will have a linear effect of the fog color added to its predefined color					
Scripts	Scripts allow actions such as object creation, object placement, etc., to be done through text commands rather than through the user interface					
Stereoscopic 3D	Allows final and real time rendering outputs to be viewed in 3D, either through use of 3D glasses or through use of a 3D display					
Maya Ultimate	This contains all the features in Maya Complete but also other permits addition/control of features such as fluid behavior, particle dispersion, and hair and cloth effects.					
	These extra features in Maya Ultimate seem to also be included in 3ds Max					
1st+3rd Person Perspectives	The programs always work in a 1st person perspective, and so using the tool as is, you could not have 3rd person.					
	However if you made a movie you could add a 3rd person avatar that moved in front of the camera to support the 3rd person view, which would also be possible in images					
Intra-object Views	The program allows views of the scene from any angle you want to set the camera to. For the 3D modeling I know you can have the program show multiple views at the same time.					
Flyover Views	With movies you can create fly-over of objects which can also be shown in program, but only with the use of animation.					
Snapshots	Snapshots cannot be done because creating a fly-over in the program would take time and a real knowledge of how to use the too--something a subject would not know or have time for					

As none of the applications provided all of the qualities desired to create this and future planned experiments, it was determined that two applications should be used. The commercial application, Maya, was selected for experimental prototyping for its ability to provide rapid, high quality rendering and object animation. However, for the experimental environment, the AFRL 2.5/3D application, Visualization of Operational Environment for Understanding and Response (VOEUR), was selected because of its accessible code and flexibility in providing both fly-through and scene snapshot navigation. Figures 2 and 3 below illustrate the exploratory scene development in Maya and VOEUR.

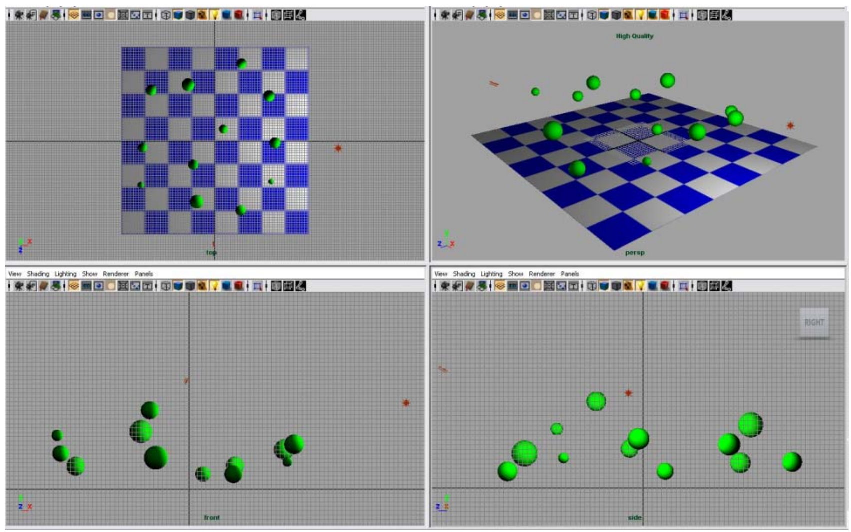


Figure 2. Maya scenes showing nodes above a gridded artificial floor

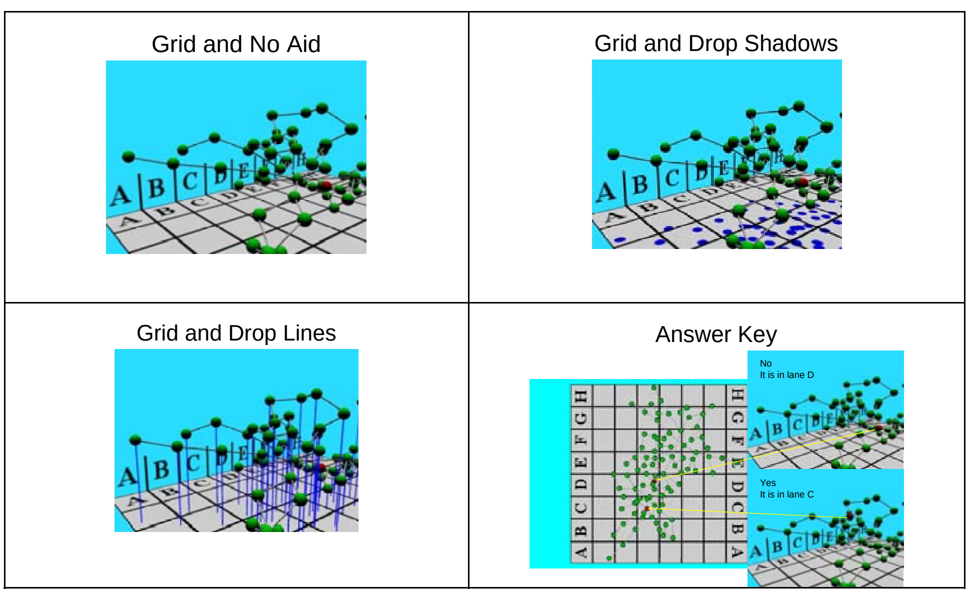


Figure 3. VOEUR scenes showing nodes augmented by drop lines and drop shadows above a labeled gridded artificial floor

2.2 Algorithm Search

The goal of the algorithm search was the identification of a layout algorithm to simultaneously optimize network use of space and network node visibility. The algorithm was determined to require the following qualities:

1. Ability to be treated as a volume (in this case a simple box) which all the nodes need to fit into.
2. Prohibits node overlap
3. Clusters grouped nodes (with shared or related attributes) tighter than other nodes
4. Avoids link overlap of both other links and of unlinked nodes

The algorithm search yielded few algorithms applicable to the layout of abstract networks in 3D. Most of the layout algorithms found applied only to 2D representations (i.e., they had no “z space” component). Of those algorithms designed for 3D, most were not appropriate for network layout. Results included algorithms based on depth cues or on depth cue perspectives; they were aligned with research into the exploitation of depth cues to facilitate robot vision and on depth cue exploitation by human viewers.

The most salient finding was a paper describing the use of an octree layout (Zhenz and Niu, 2007). An octree is a 3D construct that evenly bisects a space in each of the three dimensions, creating eight smaller sections (octants) of equal size. Each octant can then be cut again into eight smaller sections; the reduction is continued to create the level of complexity required. This algorithm was modified for use in the experimentation. The main parts of the experimental algorithm map every node in a graph into its own octant based on its closeness to other nodes. The nodes are then given a 3D position, based on the center of the octant in which they were placed when the octree “box space” size was defined. For more detail, see Appendix B.

2.3 Experimental Plan

The experimental objective is to investigate the relevance of real-world depth cues for use in visualizations of abstract data sets, (e.g. air, space and cyber networks), and to determine which application of depth cues, singly or in combination, effectively enhances depth interpretation within these abstract scenes. Visual clutter, an inescapable artifact of complex network displays, is a factor in determining effective use of display screen space; distinguishing between effective interpretive aids vs. visual clutter is a useful contribution to display design guidance. A secondary goal is to identify/evaluate parameters for mathematical formulae determining usable depth boundaries in relation to object size/scene complexity in order to determine, with respect to object interpretability, maximally effective placement of objects within 3D environments.

The experimental comparison of 2.5D and 3D visualizations of abstract data sets has three major goals:

1. Evaluate combinations of level of detail and types of interpretive aids for linear perspective and object shadowing depth cues in order to identify the dividing line between helpful visual detail and clutter.
2. Evaluate saliency of tested depth cues in moderately complex (100-node) and highly complex (300-node) abstract networks in order to identify the dividing line between helpful visual detail and clutter in networks of increasing complexity.
3. Evaluate relative interpretability of 2.5D vs. 3D network representations.

2.3.1 Equipment

__Display__***: NVIDIA Personal GeForce 3D Vision Active Shutter Glasses, and Samsung® SyncMaster ™ 2233RZ 120Hz, 22"LCD Display*** (Figure 4). The system is a bundled package that transforms a PC into a full, high definition (HD) stereoscopic 3D display. It works by synching LCD wireless active shutter glasses through an IR emitter and advanced software, to a Samsung SyncMaster 2233RZ, 120 Hz, LCD display that provides the full HD stereoscopic 3D. Subjects will view the display, seated at the prescribed viewing distance of 3 feet. During the 'true' 3D mode session, subjects will don the wireless LCD shutter glasses to view the NVIDIA display stereoscopically. For the 2.5D sessions, subjects will view the NVIDIA display without wearing the glasses or any other special device. A high performance PC workstation will execute the software for displaying the abstract network visualization trials through the NVIDIA bundled system setup. Responses will be collected using a wireless gamepad. The fluorescent room lights will be dimmed for the duration of the experiment.

Figure 4. NVIDIA Personal GeForce 3D Vision Active Shutter Glasses, and Samsung® SyncMaster ™ 2233RZ 120Hz, 22"LCD Display

__**Workstation:**__ A Hewlett-Packard high performance PC workstation will be utilized for the purpose of displaying abstract network visualizations through the NVIDIA system setup.
__**Input Device:**__ A standard wireless gamepad will be utilized for subjects to input their responses as well as advance trials.

Software: Abstract network visualization experimental stimuli were created using a modification of the AFRL-developed, 3D software environment, *Visualization of the Operational Environment for Understanding and Response* (*VOEUR*).

2.3.2 Subjects

The experimental plan calls for the testing of some 20 subjects representing both genders and ranging from 20 to 60 years of age. Subjects are being recruited from active duty, DoD and non-DoD civilians from the 711th HPW via "word-of-mouth" and email. Subjects are required to have normal or corrected-to-normal binocular visual acuity, normal depth perception, and normal color vision, determined by pre-participation vision screenings, utilizing the Bailey Lovie chart for binocular visual acuity, Stereo Fly test for depth perception, and the Ishihara test for color vision. Subjects participate in a total of four, 1 hour sessions, two sessions per display type (2.5D vs. 3D) for the study. No compensation will be provided for participation.

2.4 Experimental Description

The experimental design for this study, whose purpose is to develop more effective applications of 3D in complex abstract scenes, builds upon and extends the preliminary findings of AFRL Protocol F-WR-2008-0042-H, 'Establishing Guidelines for Effective 3D Perspective Interfaces' (reported on in Fitzhugh et al., 2009). This iteration adds more powerful computing resources, visualization software, and the application of a mathematical model for determining optimum depth boundaries relative to object size and scene complexity. In addition, the principles of Theory of Signal Detection (TSD) will be applied to the results for the purpose of exploring operator sensitivity to the sensation of depth as a function of various depth cues and display medium (2.5D vs. 3D).

Depth cues to be manipulated include interpretive aids for linear perspective, indicated by an artificial floor marked to show different depths (Figure 5) and object shadowing, indicated through drop shadows and drop lines (Figures 6).

2.4.1 Independent Variables

Seven independent variables are incorporated:

1. Linear perspective lane indicators (tick marks, lanes, grid)
2. Object depth interpretation aid (drop lines, drop shadows, no aid)
3. Network size (100 nodes vs. 300 nodes)
4. Display mode (2.5D vs. 3D)
5. View points (45 deg. vs. 90 deg.)
6. Target lanes (selected from among lanes C, D, and E)
7. Target node location truth (true vs. false)

1. Linear perspective-based lane indicators: The edges of the floor provide the converging lines that suggest linear perspective. The experimental task, determining the

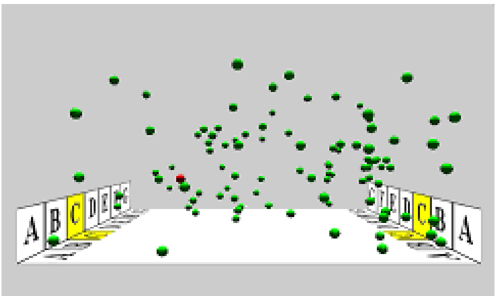
correctness of the suggested node depth, requires a means to indicate node position. The floor not only provides linear perspective, but also provides a space in which to specify depth. The experimental space is divided into depth regions, indicated by lanes marked along the edge of the floor (i.e., Lane A, Lane B, Lane C, etc.). Rather asking for absolute position, the experimental question presented to the subjects asks, “Is the target node over Lane X?” The suggested lane is highlighted to reduce visual search. Lane indication conditions are as follows (Figure 5a-c):

a. Grid (horizontal and vertical lines). Lanes are indicated on a grid. Cells formed by the grid will be set to be the same size, but due to linear perspective, will appear to diminish in area as the subject looks farther into the display.

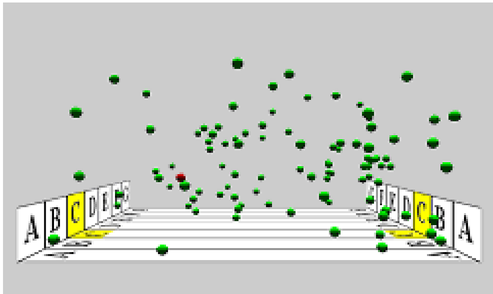
b. Lanes (horizontal lines). Lanes will be indicated on an otherwise blank floor. Lanes will be set to be the same size, but due to linear perspective, will appear to diminish in area as the subject looks into the display.

c. Tick marks (start and end of horizontal lines). Lanes are suggested by tick marks along floor edges. Again, all indications for lanes will be set to be the same size, but due to linear perspective, tick marks, and the lanes they suggest, will appear to diminish in area as the subject looks into the display.

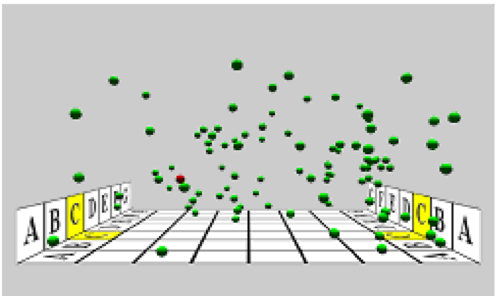
The tick mark condition serves as the control condition. It will provide minimal aid, while still providing a way to identify target depth. Floor color and line markings are consistent throughout the trials.



a: Network floor with tick marks on sides



b: Network floor marked by lanes



c: Network floor marked by gridlines

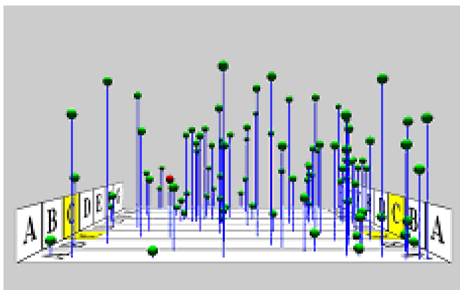
Figure 5. Artificial floor types (tick marks, lined lanes, and grid)

<u>2. Object Depth Interpretation Aids</u>: Drop lines and drop shadows can be employed in 3D scenes as interpretive aids to depth interpretation. Both aids are employed in the trials to determine whether they assist the subject to correctly respond and enhance confidence in the response. There are three aid conditions (Figures 6a-c).

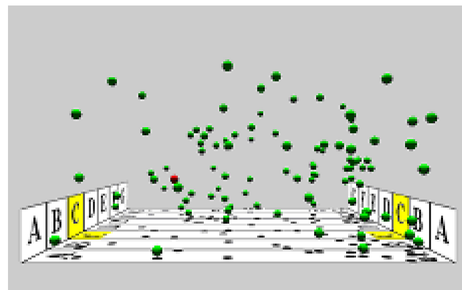
a. <u>Drop lines</u>. Lines extending from the center of each node straight down to the floor.

b. <u>Drop shadows</u>. Node shadows, cast directly below each node.

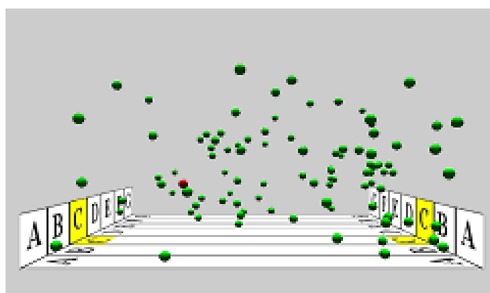
c. <u>No Aid</u>. (Null condition) The no aid, or null condition, will act as the control for the other two conditions.



a: Network with drop lines



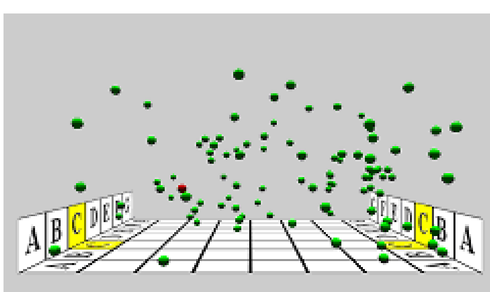
b: Network with drop shadows



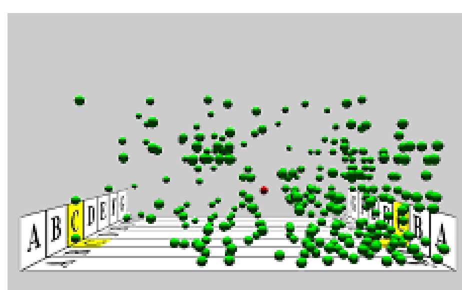
c: Network with no interpretive aid

Figure 6. Aiding conditions (drop lines, drop shadows and null condition)

<u>3. Network Size (100 vs. 300)</u>: The network size (number of nodes) simulates two levels of visual complexity, filtered and unfiltered networks (Figure7a & b). Filtered networks are represented by networks of approximately 100 nodes. Unfiltered networks are represented by networks of approximately 300 nodes.



a: Small (100-node) network

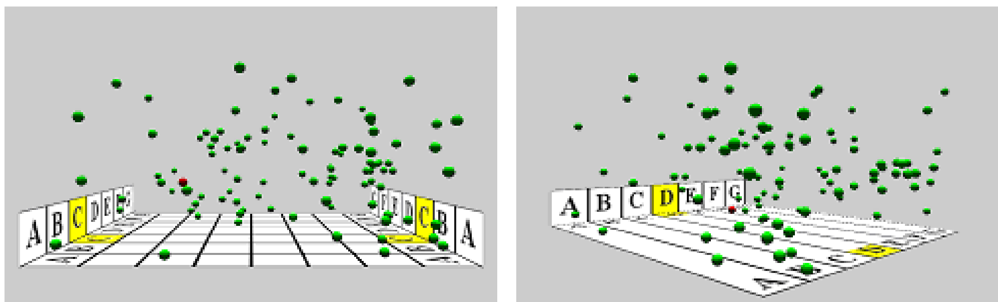


b: Larger (300-node) network

Figure 7. Network size conditions (100-node vs. 300-node networks)

<u>4. Display Mode (2.5D vs. 3D)</u>: The relative interpretability of 2.5D vs. 3D network representations is evaluated viewing abstract network visualizations on each display mode.

<u>5. View Points (45 deg. vs. 90 deg.)</u>: Each target node scene is observed from two distinct view points, positioned at 45 and 90 degrees (Figure 8a & b). This will permit the reuse of scenes while reducing the likelihood of learning effects. Presentations will be static; there will be no apparent motion in the presentation. Trials will be sequenced to avoid pairing views of the same scene.



a: Point of view 1 (90 degrees from lane markings)

b: Point of view 2 (approximately 45 degrees from lane markings)

Figure 8. Points of view (90 degrees and 45 degrees off lane markings)

<u>6. Target Lanes</u>: The effect of depth for the different lanes is examined for Lanes C, D, and E (Figure 9). The experimental question asks, “Is the target node over Lane X? (X may be either Lane C, or D, or E)?” These three lanes were selected because they fall within the midsection of the experimental area which is an area of high depth ambiguity. Each has multiple referents on each side (unlike Lanes A and B) and none could be made more visually accessible by rotating the display (unlike Lanes F and G).

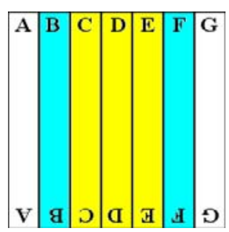


Figure 9. Target lanes

<u>7. Target node location truth (true vs. false)</u>: In keeping with the principles of TSD analysis (Wilson, 1992), half of the presentations of the target node are over the target lane (true condition); whereas, half actually are over a different lane (false condition). For each of the three target lanes (C, D, and E), the target node is either over the suggested lane (true condition) or over one or the other of the adjacent lanes (false condition). For example, Figure 10 shows a false condition, in which the target node is not over Lane C

(the indicated lane), but over Lane D. In another false condition trial, the target node will be over Lane B.

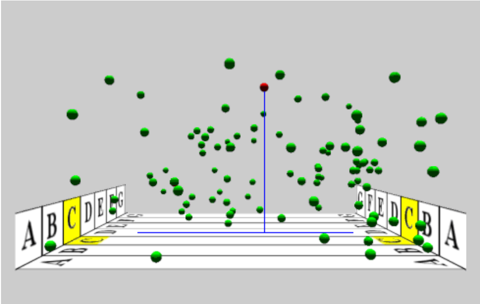
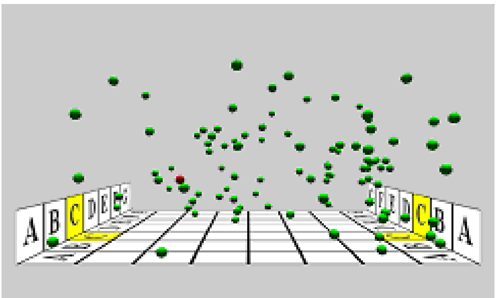
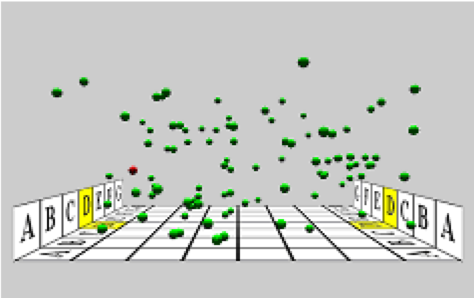


Figure 10. True depth position of target node

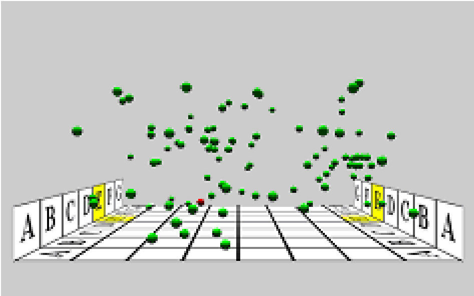
Three equivalent network node arrangements, or network configurations, (Figures 11a-c) are employed to expand the number of different visual presentations seen by the subject. These configurations will not be analyzed; they are incorporated to provide variety and reduce the likelihood of learning effects. The three network configurations will be randomly assigned across lane conditions.



a: Sample network configuration 1



b: Sample network configuration 2



c: Sample network configuration 3

Figure 11. Sample target node configurations

2.4.2 Dependent variables

Subjects are asked to judge whether the indicated depth position for a target node is correct; Hits, Misses, False Alarms, and Correct Rejections will be tabulated for each subject. Response options are framed to collect confidence levels (i.e., definitely confident, somewhat confident, not confident). Response speed also is measured.

2.4.3 Experimental Hypotheses

The design of the experiment investigates the effect of depth cues and depth cue interpretive aids on interpretation of network scenes. The use of interpretive aids to improve depth cue understanding is suggested by work done by SPAWAR using drop lines, direction vectors, and altitude markings to indicate object position relative to the ground coordinates, object heading/speed, and object altitude. While such interpretive aids would normally be anticipated to improve performance, they impose a visual cost, as they introduce clutter to already complex scenes. The experimental design is intended to differentiate among levels of clutter and determine the point of diminishing returns for the use of interpretive aids in network displays. The experimental hypotheses (H1 through H3) are as follows.

H1: Smaller network configurations will be interpreted more rapidly and accurately than larger network configurations.

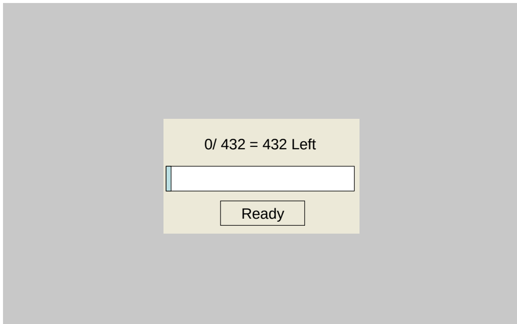
H2: Improvement in accuracy and speed will be inversely proportional to the level of clutter imposed by the addition of interpretive aids (i.e., the addition of drop lines and drop shadows, the addition of tick marks, lanes, and grids to artificial floor topography).

H3: No aiding conditions will be more difficult to interpret than aided conditions.

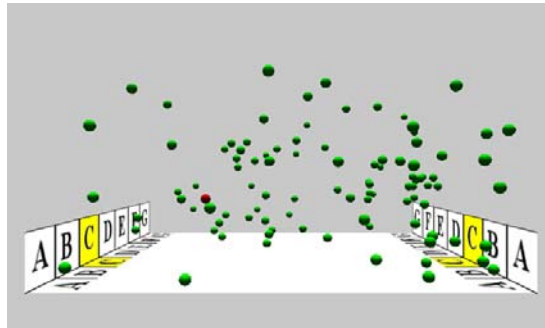
2.4.4 Presentation Method

In summary, the complete experiment incorporates a randomized, within-subjects repeated measures design: 3 (grid, lanes, tick marks) by 3 (drop lines, drop shadows, no aid) by 2 (100 nodes vs. 300 nodes) by 2 (45 deg. vs. 90 deg. view points) by 3 (Lanes C, D, and E) by 4 (True/False target lane alternatives) for each display type (i.e., experimental condition of 2.5D vs. 3D). This design yields 432 distinct presentations of experimental conditions for each test session. The experiment is being conducted in four sessions, lasting approximately one hour each. Each subject will complete a total of 1728 trials across the four test sessions (two sessions in each display mode). Presentations for each display mode will be counter-balanced across the four test sessions. Display mode presentation is being randomized across subjects. For each test session, the order of trial presentations will be randomized across subjects.

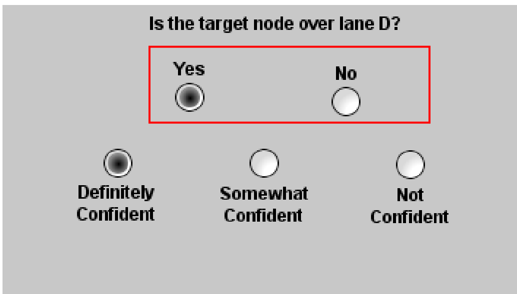
Each session begins with a practice session on the NVIDIA to accustom the subject to the experiment and the display mode. The trials will begin with a ready screen. Subjects press a button on a gamepad to advance to the network presentation. The network presentation remains onscreen for a maximum of 5 seconds. After 5 seconds, the network display is replaced by a question screen. The subject responds "Yes" or "No" to the experimental question (Is the target node over Lane X?) and will choose among 3 confidence levels (Definitely Confident, Somewhat Confident, Not Confident). The experiment will not advance until the subject responds. See Figure 12a-d.



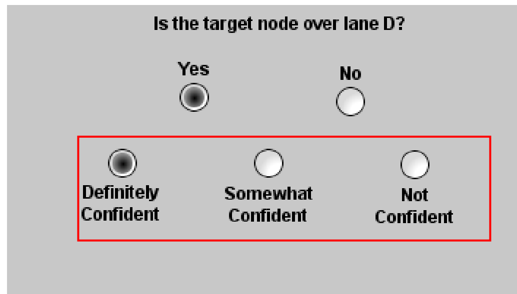
a. Initial "Ready" screen



b. Network Presentation



c. Response prompt



d. Confidence prompt.

Figure 12. Example test communication screens

2.4.5 Data Analysis

Data analysis will utilize TSD methods as well as standard analysis of variance (ANOVA) procedures. The study will utilize TSD for the purpose of exploring operator sensitivity to the sensation of depth as a function of various cues and display medium (2.5D vs. 3D). TSD assumes the decision maker is not a passive receiver of information, but an active decision maker who makes difficult perceptual judgments under conditions of uncertainty. TSD quantifies the subject's decision criterion as well as their visual sensitivity making it possible to tease out observer response bias which will validate the assessment of the applicability of 2.5D and 3D for visualizations of abstract data sets (Wilson, 1992).

Data collection will employ TSD analytical methods for the purpose of obtaining subject confidence levels with the overarching goal to tease out subject sensitivity to depth as a function of depth cues and display medium. Responses will fall into four categories. These categories will be analyzed with d', beta and a_g calculated for observer sensitivity, as well as generating receiver operating characteristics (ROC) curves to illustrate the sense of stringency of the subjects' internal response criteria or bias for depth cues and display medium:

	Correct Response	**Incorrect Response**
Accept	Correct Acceptance/Hit	Incorrect Acceptance/False Alarm
Reject	Correct Rejection	Incorrect Rejection/Miss

The experimental question to which subjects respond is: “Is the target node over lane X?” (X will be C, D, or E). Lanes are labeled on the floor and represent a depth region identified by a letter (e.g., Lane C, D, or E). In order to obtain subject confidence levels, response options are phrased as follows:

1		2		3
o		o		o
Definitely confident		Somewhat confident		Not confident

2.4.6 Data Collection

The following information is being collected for each trial and recorded on an HP Workstation PC connected to the NVIDIA display:

1. Trial number
2. Trial order
3. Floor type
4. Visual Aid
5. Network size
6. Network
7. Target node
8. View point
9. Correct response
10. Subject response
11. Answer correct? (Coded as Hit, Miss, False Alarm, or Correct Rejection)
12. Time to respond

The following information is being collected and recorded via the HP PC workstation for each session:

1. Subject identifier
2. Session/Display Mode (2.5D or 3D)
3. Total time to complete the session

2.4.7 Training

Subjects receive a briefing that explains the purpose and conduct of the experiment. A series of practice questions are provided for each display mode to ensure subjects are comfortable with equipment and procedures. The training briefing may be found in Appendix A

2.4.8 Software Description

The experimental platform employs custom software created specifically for this experiment using AFRL’s VOEUR application code base. VOEUR is built on the AFRL JVIEW 3D engine; the software leverages the JView API for 3D graphic rendering. Special algorithms have been created or modified to generate the network configurations in their several sizes and to optimize the node layout for visibility. Tailored algorithms

control target node selection as well as drop shadow and drop line positioning and the placement and form of the various versions of the artificial floor whose edges provide the linear perspective. Modified randomization algorithms control the presentation of the trials within each session. Full randomization of individual trials within sessions is not desirable in this case, as the experimental intent is to avoid rather than investigate learning effects; therefore, sequences that might provide learning opportunities are barred. Within the 2.5D and 3D display modes, the software randomizes the order of session presentations.

The session presentations are movie files; upon completion of the five-second movie, the subject controls the pace of the response. Responses are indicated using radio buttons. Individual responses, speed, accuracy, distance from correct selection, presentation details and ground truth are collected for each trial and stored in an Excel spreadsheet. See Appendix B for more detail.

2.4.9 Pretests and findings

Several sets of pretests were run to ensure the efficacy of the software, the suitability of the experimental design, and to determine whether results were aligned with experimental goals. Initial concerns centered on subject confusion and subject fatigue as potential confounding factors. During prescreening of full experimental sessions two subjects noted visual fatigues. Subsequent prework halved the presentation; two subjects completed the shorter 2.5D session and found no issues with confusion or fatigue. The second trial ran two subjects in single tests of 2.5D and 3D sessions. Subject fatigue did not appear to be a factor. Issues subjects raised regarding training and presentation were mitigated through adjustments to the experimental presentation. Finally, two subjects ran through complete sets of 2.5D and 3D sessions (four sessions total) without incident.

The last pair of pretests was used to set up the statistical analysis. Separate preliminary analyses were set up and run for proportion of hits, proportion of false alarms and response time dependent variables and display mode, floor type, visual aid type, network size, target height and subject independent variables using the (preliminary) data from two pretest subjects. An “order” independent variable was included for the “response time” dependent variable analysis only. A sample ROC curve was generated from preliminary data, focusing on the effects of the floor markings on accuracy.

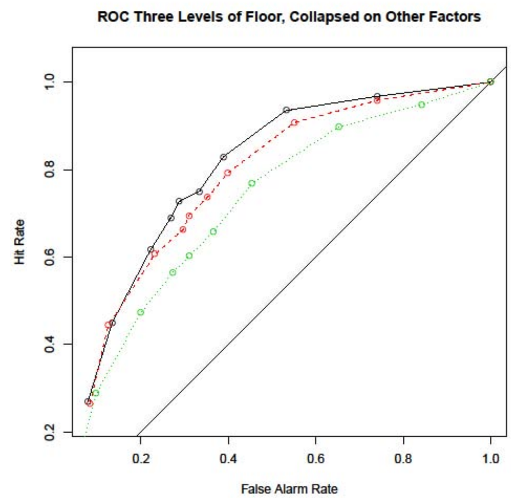


Figure 13. Sample ROC curve generated from 2 pretest subjects' data

The analysis assumes a random subject effect, with the other independent variables having fixed effects. Only two and three-way interactions were included in the model. The analyses developed intended to use the Greenhouse-Geisser epsilon to adjust F tests for the sphericity assumption, if necessary. [For more information, see *Design and Analysis* by Geoffrey Keppel and Thomas Wickens.]

Once the pretests were completed, the analytic set-up was expanded to include two sessions per display mode, for a total of four sessions per subject. This included labeling data by session. The analytic files were provided to the government to use as templates in analyzing the actual experimental results.

3.0 Experimental Results

Although the experimentation has not been completed at the time of this writing, and no experimental results are available, the subjects who have completed all experimental sessions report no noteworthy fatigue. Initial concerns that the use of 3D glasses, the visual demands of the 3D presentations or other unanticipated factors relating to 3D technology would generate complaints appear to be unfounded. Subjects are able to reliably distinguish foreground targets vs. background targets (i.e., targets appearing nearer the viewer vs. targets appearing farther from the viewer). However, actual subject accuracy is often at variance with subject expectation. Distance above the artificial floor appears to be a factor in both speed and accuracy of position judgments.

Subjects have expressed a preference for smaller networks, which limit the number of objects cluttering the scene. Within smaller networks they have expressed a preference for interpretive aids, although response is split on preference for drop shadows vs. drop lines. The drop lines are noted as making judgment easier if they are not occluded and if the level of visual clutter is not high, but in crowded scenes, some subjects preferred the drop shadows. Several subjects have requested both drop lines and drop shadows, as well as the ability to toggle the interpretive aids on and off.

4.0 Discussion

The importance of investigations into the use of 3D in both realistic and abstract scenes is underscored by the variability of its empirically shown effectiveness as a display medium. Work done by the research teams in SPAWAR SSC San Diego (e.g., St. John, Cowen, Smallman, & Oonk, 2001; Smallman, St. John, & Cowen, 2002) illustrates the limitations inherent in 3D visualizations, as their investigations showed 3D provides superior utility in qualitative decisions (e.g., shape recognition) but inferior utility in quantitative (e.g., relative position judgments). Seeking a neurophysiological basis for their observations, the group conducted experiments that found "pervasive" cognition-based perspective interpretation errors—the brain computes depth and width compression at comparable rate, whereas, within the geometry of a simulated scene, depth actually compresses significantly more rapidly than width. The visual processing system does not fully account for differences in actual vs. simulated 3D, making it a less than accurate depth assessor, whose error magnitude increases with scene depth.

Any difficulties in accurately interpreting object depth and altitude, both of which are examined in this study, certainly relate to airspace management visualizations but equally, relate to comprehension of and movement within abstract network displays. Coding schemes, are frequently used to increase the immediately accessible visual information displayed in abstract networks; however, should they rely on clustering in 3D-space or on relative distance from the viewer, coding could be less rather than more informative due to potential visual confusion. Atmospheric effects employed to indicate depth are anticipated to negatively impact color recognition and even shape coding, as the atmospheric conditions (as well as scene light sources) darken distant objects and soften their outlines. The same visual challenges can be expected in both simulated and real 3D. This is less of an issue when the scene to be interpreted is realistic as opposed to abstract, as the ability to move through and around a scene can clarify distant objects in realistic scenes. However, abstract networks typically have far fewer interpretable cues (recognizable landmarks) to aid rapid and sure navigation. Inclusion of natural cues such as perspective and horizon lines and object shadows and artificial cues such as drop lines and header lines can reasonably be expected to help with depth interpretation, if not with landmarks, but can also be expected to add clutter to an already complex visual representation.

The current experimentation seeks to find methods to mitigate the clutter imposed by visual cues by identifying a point of diminishing returns. The experiment seeks evidence for an optimal set of cues—providing reasonable interpretive power without obscuring important information. The results analysis should yield interesting results regarding subject ability over time. The joint use of topographical cues (the marked artificial floors) and depth interpretation aids are anticipated to interact with scene complexity. Experimenters also are interested in and anticipate that visual fatigue effects may be found. The length of experimental sessions and the visual burdens imposed during the trials are deliberate. Working with 3D scenes over time, while becoming less onerous as 3D technology evolves, may still be found to induce unacceptable levels of fatigue.

5.0 Recommendations

There is insufficient evidence to date to determine how well 3D technologies will support visual comprehension of complex multidimensional data sets. More experimentation in the comparison of 2D, 2.5D and 3D medium as well as in the use of interpretive aids, the limits to coding schemes, and methods for navigation support should provide better guidance for visualization design.

A particular issue that should receive further attention is that of navigation through 3D abstract networks. Wickens (2000) explored the effect of field of view (tethered/exocentric and immersed/egocentric views) in 3D vs. coplanar display navigation and scene interpretation. He noted visual costs associated with line of sight ambiguities for both 3D displays as well as keyhole view limitations and high scanning costs for egocentric viewpoints. Navigation through abstract representations, devoid of memorable landmarks, is anticipated to pose challenges to users, whether they operate from exocentric or egocentric fields of view. Additionally, it is anticipated that abstract scenes will provide few cues to maintain a sense of position in space and a memory for traversed paths. Phase Two of the current effort is intended to explore, separately and in combination, three navigation aiding methods: 1) inset flyover views, 2) user-collected scene snapshots, and 3) controlled, incrementally constrained movements in the horizontal and vertical planes. The effort also is intended to collect evidence for guidance in the appropriate use of field of view—in what navigation tasks, within abstract scenes, is it preferable to use an egocentric vs. an exocentric perspective and does it enhance performance to allow users toggle between the two. The Phase Two design will employ lessons learned in the conduct of Phase One and will be conducted and reported subsequent to Phase One completion.

These suggestions only begin to detail the experimentation that may usefully be pursued to ensure 2.5D and 3D technologies are appropriately and effectively employed. The attractiveness of the new technologies lures the user; but does it benefit the work it is intended to support? And if, in abstract network interpretation and navigation, it does enhance performance, then how may scenes be constructed to facilitate rapid and accurate judgments in support of task completion? Resolutions of these questions drive the AFRL 3D technology research series.

6.0 Conclusion

This report describes the second in a series of AFRL-led research efforts to develop guidance for the use and design of 3D visualizations. A 2008 survey of 3D research trends showed a much heavier emphasis on resolution of technological constraints and proofs of concept for new technologies than on user experience and human performance. 3D visualizations may be employed in diverse roles and in multiple domains; established research threads followed the use of 3D in robotics, in physiological visualizations, in airspace management and navigation, in sea lane management and navigation, in aerospace and naval mission rehearsal and in urban terrain depiction for both security management and mission rehearsal. However, as 3D technologies mature, research opportunities remain open-ended—all that might need to be known is not yet known. Research threads investigate the efficacy of 3D as a display medium for large data sets, especially for

visualizing the contents of data repositories and envisioning search results—which may number in the millions. Yet another research thread investigates the employment of 3D to manage projects and project data, using personal desktops and filing systems as a metaphor. Some research also explores issues in 3D presentations depicting multidimensional complex data sets. However, most of it is focused on the quantitative analysis of data sets themselves, whether displayed in geometric shapes or in 3D-enhanced charts. However, few efforts explored the use of 3D to display, understand, and manage such abstract concepts as social networks, where each node is packed with information, links depict important relationships, and identified pathways may support predictive reasoning. The basic research that supports such applied efforts, such as the effort this report describes, is being done—but it is in its infancy. Command interest in early employment of potentially useful technologies, especially 3D technologies, suggests continued research into effective use of 3D in military applications is critically important to the development of useful, usable tools whose visualizations speed the accurate accomplishment of tasks across work disciplines.

7.0 Acronyms

AFRL	Air Force Research Laboratory
ANOVA	Analysis of Variance
COTS	Commercial-Off-the-Shelf
GOTS	Government-Off-the-Shelf
ICER	Information Operations-Cyber Exploitation Research
RHCVZ	Human Effectiveness Directorate, Warfighter Interface Division, Battlespace Visualization Branch
ROC	Receiver Operating Characteristic
TSD	Theory of Signal Detection
VOEUR	Visualization for Operational Environment Understanding and Response

8.0 References

Andre, A., & Wickens, C. (1995). When users want what's not best for them. *Ergonomics in Design*, (October), 10-14.

Aragon, C. (2004). Usability Evaluation of a Flight Deck Airflow Hazard Visualization System. *Proceedings of the 23rd Annual Meeting of the Digital Avionics Systems Conference*, pp. 4.B.2 - 41-11 Vol.1.

Barco. (2009). Barco and Airways New Zealand to jointly develop arrival and departure management solution. Accessed online 1 June 2010 at http://www.barco.com/en/pressrelease/2406/en

Bardel, William H. Depth Cues for Information Design. [Master's Thesis]. Pittsburgh PA: Carnegie Mellon University School of Design.

Bemis, S., Schiller, E. & Ko, H. (2001). Information Presentation on 3-D Perspective Displays. *Proceedings of the Human Factors and Ergonomics Society 45th Annual Meeting*.

Boyd, D. (2000). Depth Cues in Virtual Reality and the Real World: Understanding Individual Differences in Depth Perception by Studying Shape-from-shading and Motion Parallax. [Thesis]. Brown University, Computer Science Department.

Dixon, S., Fitzhugh, E., & Aleva, D. (2009).Human Factors Guidelines for Applications of 3D Perspectives: A Literature Review. *Proceedings of SPIE*, Vol. 7327, 732708, pp. 1-11.

Fitzhugh, E., Dixon, S., Aleva, D., Smith, E., Ghrayeb, J., & Douglas, L. (2009). Toward the Establishment of Design Guidelines for Effective 3D Perspective Interfaces. *Proceedings of SPIE*, Vol. 7327, 73270K, pp.1-11.

Hand, R. (22 April 2010). Air Force to Give All Recruits Second Life Avatars. *Vizworld.* Accessed online 1 June 2010 at: http://www.vizworld.com/2010/04/air-force-give-recruits-virtual-world-avatars/

Havig, P., McIntire, J., Dixon, S., Moore, J., & Reis, G. (2008). Comparison of 3D Displays Using Objective Metrics. *Proceedings of SPIE*, Vol. 6956, p. 69560.

http://www.metavr.com/products/vrsg/vrsgoverview.html

Jumper, J. (2003). Speech by General John P. Jumper. Given at the Air Force Association (AFA) National Symposium, Orlando, FL, February 13, 2003. Accessed online 1 June 2010 at http://afa.org/aef/pub/jump203.asp.

Lawson, B.D., Graeber, D.A., Mead, A.M., & Muth, E.R. (2002). Signs and Symptoms of Human Syndromes Associated with Synthetic Experiences. In K.M. Stanney (Ed.) *Handbook of Virtual Environments: Design, Implementation, and Applications*. Mahwah, NJ: Lawrence Erlbaum Associates.

Liao, M. & Johnson, W. (2001). Effects of Transformed Visual-Motor Spatial Mappings and Droplines on 3-D Target Acquisition Strategy and Performance. *Proceedings of the Human Factors and Ergonomics Society 45th Annual Meeting*, pp. 1367-1371.

Ma, K. L. (2007). VisFiles - The Next Surge of Visualization Research. *Computer Graphics Quarterly, 41*(3). Accessed online 1 June 2010 at http://www.siggraph.org/publications/newsletter/volume/visfiles-the-next-surge-of-visualization-research

Naikar, N. (1998). Perspective Displays: A Review of Human Factors Issues. Report No. DSTO-TR-0630. Melbourne, Victoria: Defence Science and Technology Organisation Aeronautical and Maritime Research Laboratory.

Smallman, H. & St. John, M. (2005). Naïve Realism: Limits of Realism as a Display Principle. *Proceedings of the Human Factors & Ergonomics Society 49th Annual Meeting—2005*, pp. 1564-1568.

Smallman, H., Cook, M., Manes, D., & Cowen, M. (2007). Naïve Realism in Terrain Appreciation. *Proceedings of the Human Factors & Ergonomics Society 51st Annual Meeting—2007*, pp 1317-1321.

Smallman, H., Schiller, E. & Cowen, M. (2000). Track Location Enhancements for Perspective View Displays. Technical Report No. 1847. San Diego, CA:SPAWAR.

Smallman, H., Schiller, E. & Mitchell, C. (1999). Designing a Display for the Area Air Defense Commander. Technical Report No. 1803. San Diego, CA: SPAWAR.

St. John, M. & Cowen, M. (1999). Using 2-D vs. 3-D Displays: Gestalt and Precision Tasks. *Proceedings of the Human Factors and Ergonomics Society 43rd Annual Meeting—1999*, pp.1318-1322.

St. John, M., Cowen, M. B., Smallman, H. S., & Oonk, H. M. (2001). The use of 2-D and 3-D displays for shape understanding vs. relative position tasks. *Human Factors 43*, 79-98.

Smallman, H. S., St. John, M., & Cowen, M. B. (2002). Use and misuse of linear perspective in the perceptual reconstruction of 3-D perspective view displays. *Proceedings of the 46th Annual Meeting of the Human Factors and Ergonomics Society*—2002, pp. 1560-1564.

St. John, M., Smallman, H. & Cowen M. (2000). Designing for the Task: Sometimes 2-D is Just Plane Better. *Proceedings of the IEA/HFES 2000 Congress*, pp. 3-207.

Ware, C. (2003). Designing with 2 1/2D attitude. University of New Hampshire, Durham, NH: Chase Ocean Engineering Lab, Center for Coastal and Ocean Mapping—Data Visualization Research Lab. Accessed online at May 21 at http://ccom.unh.edu/vislab/PDFs/Design2.5D.pdf

Weiskopf, D. & Ertl, T. (2002). Generic and Real-Time Color-Based Depth-Cueing Scheme. Report TR-2002/08. Universität Stuttgart, Fakultät Informatik. Accessed online 1 June 2010 at http://wwwvis.informatik.uni-stuttgart.de/depthcue

Wickens, C. & Carswell, C. (1995). The Proximity Compatibility Principle: Its Psychological foundation and relevance to display design. *Human Factors & Ergonomics Society*, 37, 473-494.

Wickens, C. & May, P. (1995). Terrain Representation for Air Traffic Control: A Comparison of Perspective with Plan View Displays.

Wickens, C. (2000). The When and How of Using 2D and 3D Displays for Operational Tasks. *Proceedings of the IEA 2000/HFES 2000 Congress*, pp. 3-403–3-406.

Wilson, Denise L. Theory of Signal Detection and Its Application to Visual Target Acquisition: A Review of the Literature. AL-TR-1992-0083. Armstrong Laboratory, Wright-Patterson AFB, OH (1992).

Zheng, J. & Niu, J. (2007). Unified Mapping of Social Networks into 3D Space. *Proceedings of the Second International Multisymposium on Computer and Computational Sciences—2007*.

Appendix A: Training Materials

3D Visualization of Abstract Data Sets

Phase One Experiment

Subject Training

Experimental Goal

- We are looking at how depth is perceived and interpreted in 2.5D and 3D displays.
  - Relative size and linear perspective (the sense that objects in a scene both grow *smaller* and *closer together* as one approaches the visual horizon) are two primary depth cues we are investigating.
  - We are using drop lines and drop shadows, as well as several ways of marking depth lanes, to see when and how much those aids help you determine the depth of the objects you view.

Experimental Conditions

- This experiment will be conducted in two separate sessions.
  - In one session, you will look at link and node networks in a 2.5D environment.
  - In the other session, you will wear special glasses to look at the same networks in a 3D environment.
- You will try to identify the target node's depth location.
  - The experiment will show multiple networks viewed from two different angles (45° and 90°).
  - Networks will have two levels of complexity—either 100 or 300 nodes.
  - Test images will have depth lanes indicated by tick marks, lines or grid marks.
  - Test images will use drop lines, drop shadows, or no cue to help you identify the correct depth lane.

Equipment

- NVIDIA personal GeForce 3D Vision Active Shutter Glasses
- Samsung® SyncMaster™ 2233RZ 120Hz, 22"LCD Display
- Logitech cordless Rumblepad 2 game pad

Navigating the Experiment

Shoulder buttons move you left or right

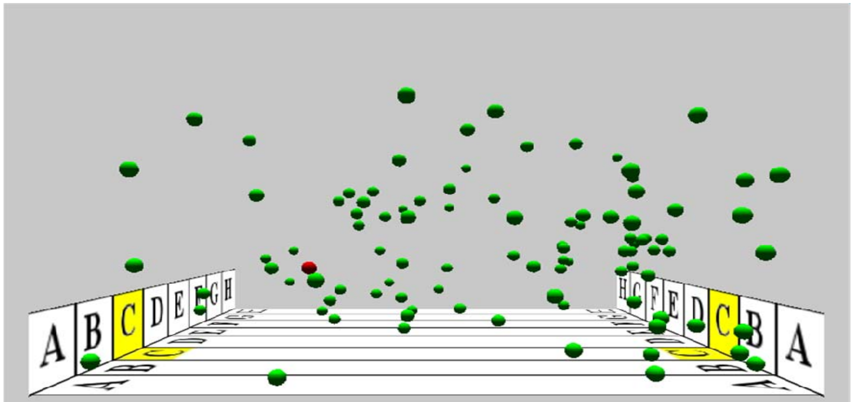
Top button returns you to change your answer

Bottom button moves to next screen

How it works…

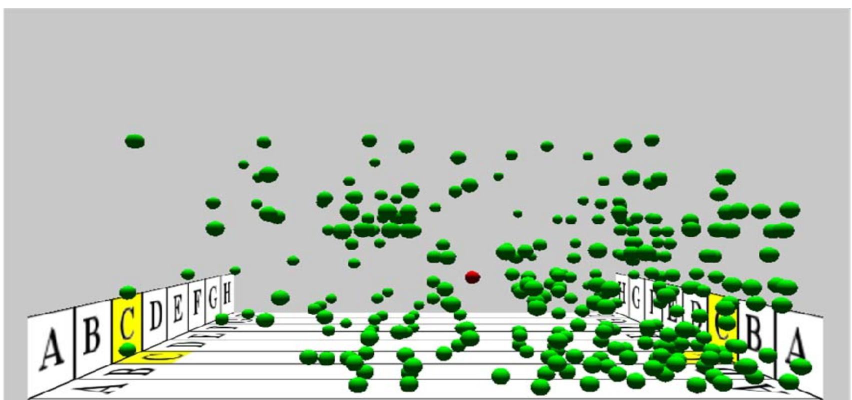
- Do the best you can. You are not expected to get every question right.
- Speed and accuracy will be collected only to see which aids are more helpful.
- You have **5 seconds** to view the 3D image and respond; if you don't begin to respond within **5 seconds**, that trial will move to the end of the queue. The experiment will continue until you have responded to each trial image.
- Once you have begun to respond, you must identify both the accuracy of the suggested depth lane and your confidence in your answer before the next image will be shown.
- You may take a break whenever you need one while on the ready screen.
- You may withdraw from the experiment at anytime by informing the administrator you want to drop out.

Example Small Network



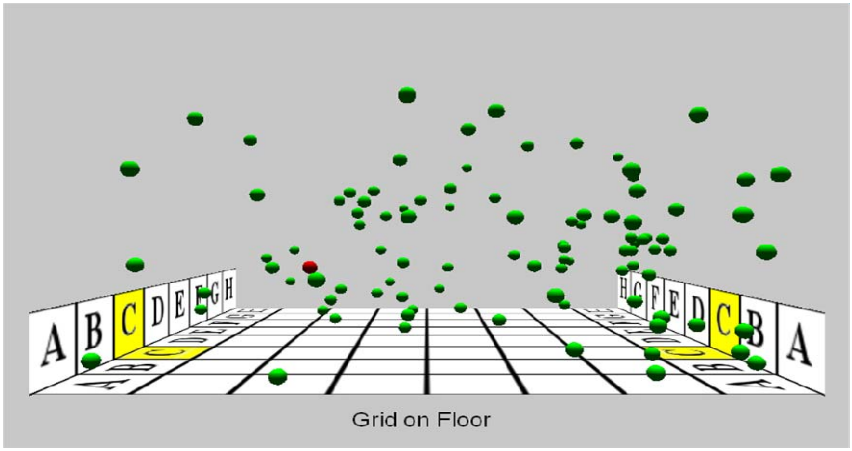
Small Network = About 100 nodes

Example Large Network

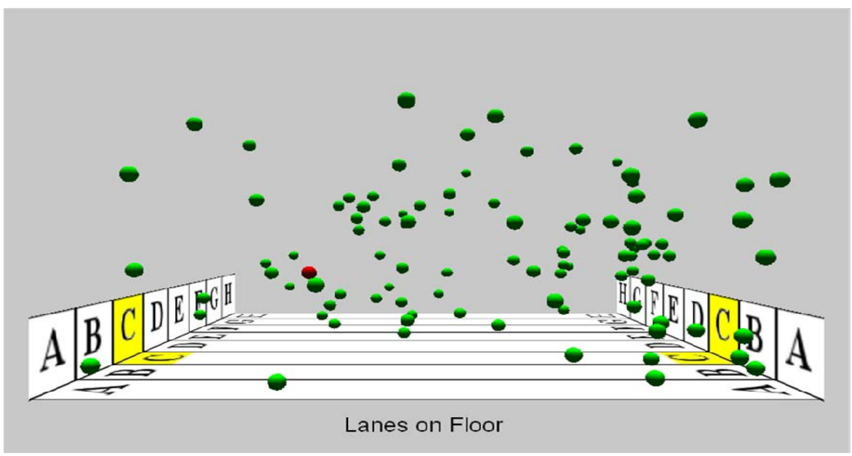


Large Network = About 300 nodes

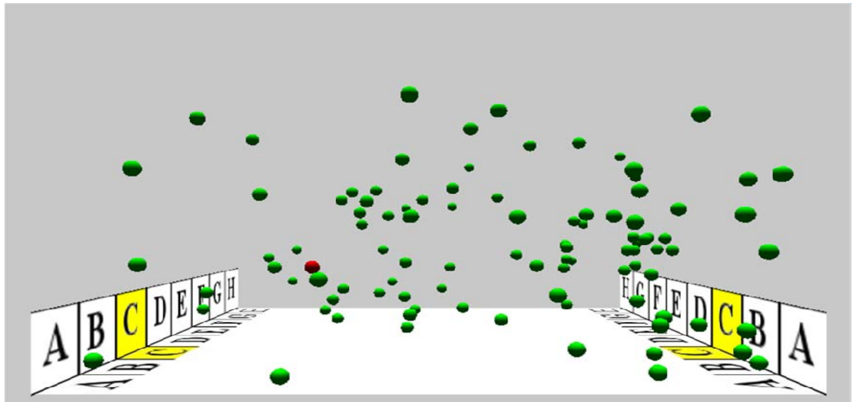
Grid-marked Depth Lane



Line-marked Depth Lane

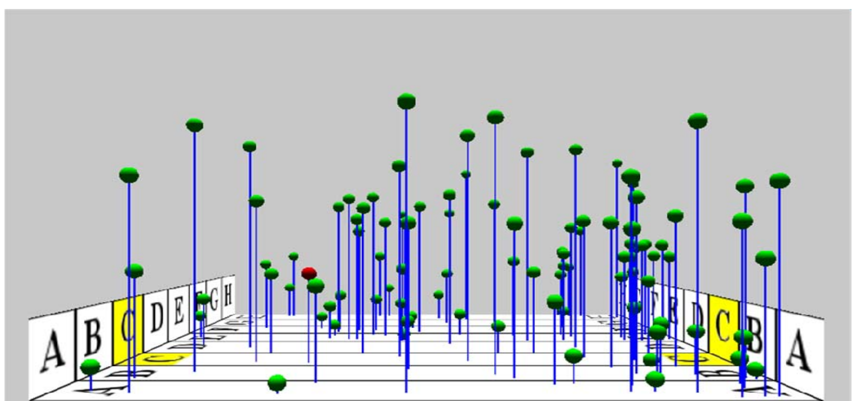


Tick-marked Depth Lane



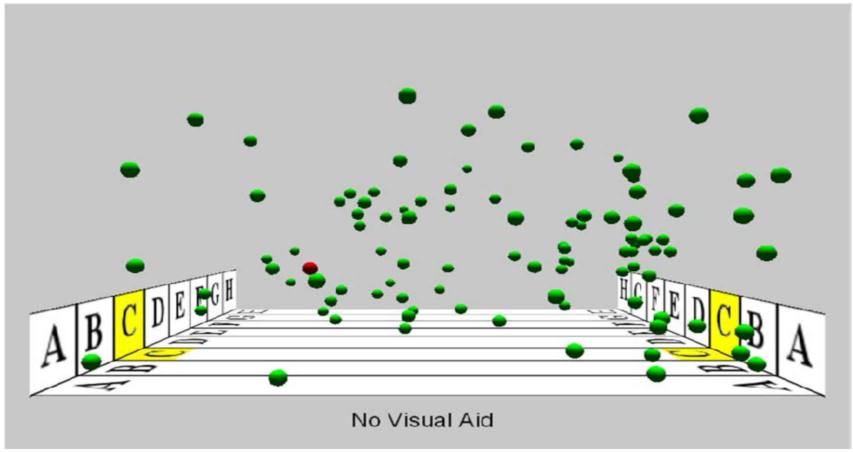
Tick Marks on Floor

Drop Lines Marking Position

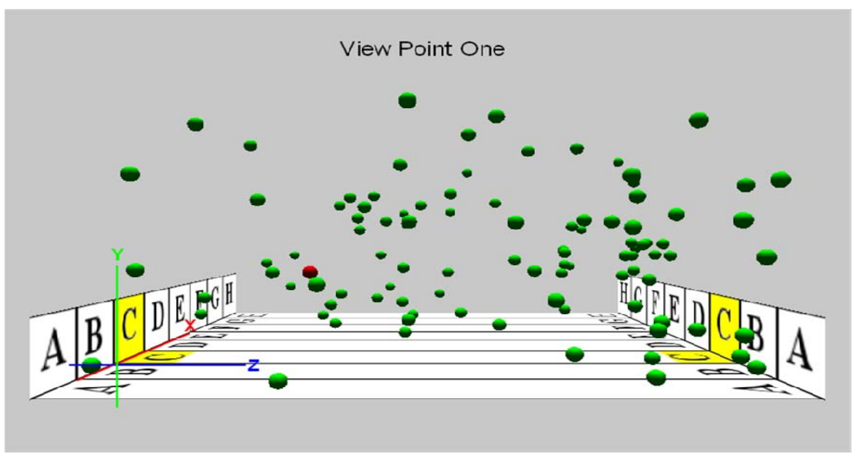


Drop Lines Visual Aid

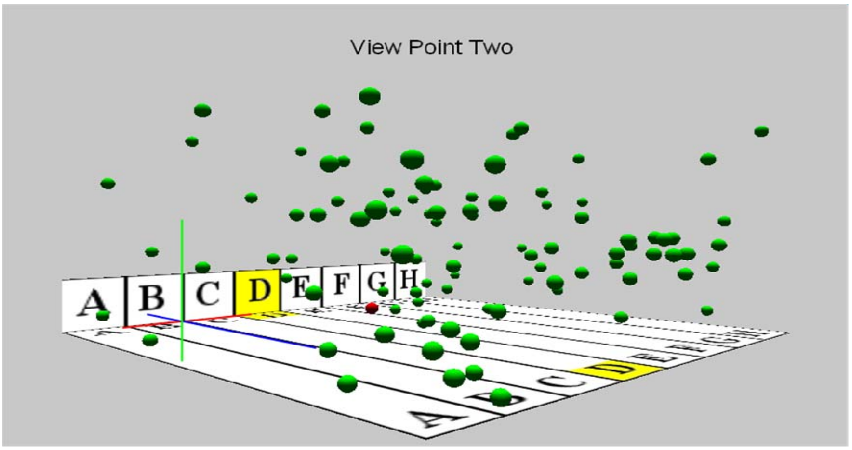
No Aid Marking Position



View rotated 90° degrees



View Rotated 45°



Appendix B: Detailed Software Description

Creating the Experiment Platform

To lower development time the experiment platform (EP), used in the 2008 experiment, was used as the starting point for this experiment's platform. The original EP used Visualization of the Operational Environment for Understanding and Response (VOEUR) as its code base. Since VOEUR evolved since the 2008 EP was first merged with the changes to VOEUR which includes the switch to Java 1.6 and a newer version of the 3D graphic engine (JView) is being used.

A main change that had to be made first to the EP was to have it run in full screen mode so that 3D displays can be used in the experiment. This task required pulling out the use of the Eclipse's RCP since it needed a simple Java application instead of a full blown windowed application. The Eclipse windows environment was kept around in a specific version which is used for handling the results as well creating data from the experiment. So the EP for this experiment makes use of two separate programs which share the same code base.

Like the 2008 EP this EP makes use of the layered view from VOEUR for displaying the graphics for the experiment. Also when a subject uses the EP a data file containing information about all the interactions (which could be watched like a movie, but this part of the EP was never implemented) as well as Excel data.

Shadows

For design of the experiment required the use of shadows as the drop shadow visual aid. Since shadows are not native to OpenGL (the underlining 3D graphic API used by JView) or JView the EP had to extend JView so that shadows could be in the experiment. The shadows were implemented with the use of the basic shadow mapping algorithm which also made use of hardware acceleration by utilizing something called a "P Buffer". This basic implementation allowed shadows to be cast on everything, but also had some graphical artifacts causing the scenes to not look correct. Since the drop shadows were only needed for the floor (so shadows were not needed to be shown on the nodes themselves) simplifications of the algorithm were allowed which greatly improved the shadows. The simplification made it so only the floor would display shadows and only the nodes would cast shadows.

Input

In the 2008 experiment the subject uses the mouse to click a node which represents their answer. In this experiment the subject had to answer a Theory of Single Detection (TSD) question. A gamepad was setup for allowing the subject to move through the experiment and select their answers from an on screen menu. Adding support for the gamepad inside the EP was very easy to do because of the support that already existed in the code base. After setting up the required actions in the code any single button press on the gamepad could be mapped to the predefined actions. The underlining technology behind the gamepad support is a Java based API called JInput.

3D Display Support

At the start of the EP development the most recent release of JView (1.6) did not have support for 3D with use of shutter glasses, so a new version (an unfinished version of JView 1.7) was provided by the JView team which included support for the glasses. In JView, to use the shutter glasses implementation a specific type of Graph3D object (the main component which is used to display the rendered scene) had to be used and there is no on/off switch for the true 3D mode. So the EP had to either startup in 2.5D mode or 3D mode, which is selected through a dialog boxes prior to going full screen. Since the 2.5D shadows implementation extended the Graph3D object, a similar extension was needed for the shutter glasses version of the Graph3D object as well. To avoid duplicate code the main part of my shadow implementation was moved out of the Graph3D object and into a helper object that my two extended Graph3D objects made use of. The two Graph3D extended classes also implemented same interface which provide access to the required shadow related functionality.

Randomization

After running some test subjects with a small test experiment an issue about the randomness of trials became apparent. For example if you saw a trial that you considered to be really easy (drop lines with lanes) and then the very next one you saw looked exactly the same, but with a harder setup (no aids) the subject could have an easy time using the information from the previous “easy” trial to answer the current “hard” trial question instead of just relying on what is seen just for the current question as intended.

To help prevent this from happening in the experiment, the randomizing algorithm was changed to avoid two trials that look the same from being shown right after each other. In the new algorithm each trial is given a string (Using the Random Like tag in the experiment movie format) based on what the scene looks like (based on the network size, configuration and view point). The trials are randomly shuffled like normal, but then the list is fixed by re-positing trials so they are not next to others that look like them.

For a simple example of how this works assume that there are nine trials which are grouped into three, three trial groups, where each trial in a group are similar to each other. Before the random shuffling the groups might looks something like this:

A	A	A	B	B	B	C	C	C

Let’s assume that the random shuffle places the trials in this order (all we care about right now is the likeness of the trials):

C	A	B	A	A	C	B	C	B

The randomization would then fix the right after the other problem by changing the order to:

C	A	B	A	C	A	B	C	B

Creating the Experiment Sessions

Creating Trial Layers

For creating content for the experiments session the 3D layers view from the 2008 experiment platform (EP) was utilized so that every specific network used in the experiment was given its own layer in. This way with the help of the experiment movie file it was made so the subject only saw one layer at a time so the given condition combination could be tested just by using a single layer and camera view setup for that network.

Generating Networks

Networks can be generated randomly or with simple rules to be in a specific type of structure. However, since the main types of networks that were interested in for this experiment for social networks, just using a random network wasn't the most desirable idea. To add more real world situations to the networks the idea to use search results from Google was decided on for generating the networks.

The Google AJAX Search API (http://code.google.com/apis/ajaxsearch/) which uses HTTP requests to return search results similar to Google's normal website interface (but ads and special item callouts are not included) was used for accessing Google through code, but there were a few problems with using it. For one it didn't give a lot of detailed information about any given item such as a score, keyword count or thing like that. A second problem was that it would only return a max of 60 results (which come in either pages of size four or eight).

Since search results do not come in a tree or graph form by default the results had to be turned into a graph by use of an algorithm. Figure 14 shows an example of the structure that of the network which was decided to be created by the algorithm using the search results. The main root (can be thought of as something like: Google Search) has children of the keywords used in the search and node which has children for each page of results. The pages are not based on the pages from using the AJAX search, but based on the order the results are back with each page having the same size (ten is used since that is the default page size from Google.) Each page node is numbered and is used as a way to specify the order sites were returned by Google in (Page 1 is the first results, and Page 2 is the next result set etc.) Off of each page node are copies of the keyword nodes which are linked to the main keyword node of the given keyword. These page keywords are only in place to place a limit of links that a single node can be a part of. Otherwise a main keyword in a network with 100 nodes could have links to every site node (maybe around 80 links total) which would probably cause problems for a layout algorithm to layout of the network in a nicer way then if there was far less links for a given node.

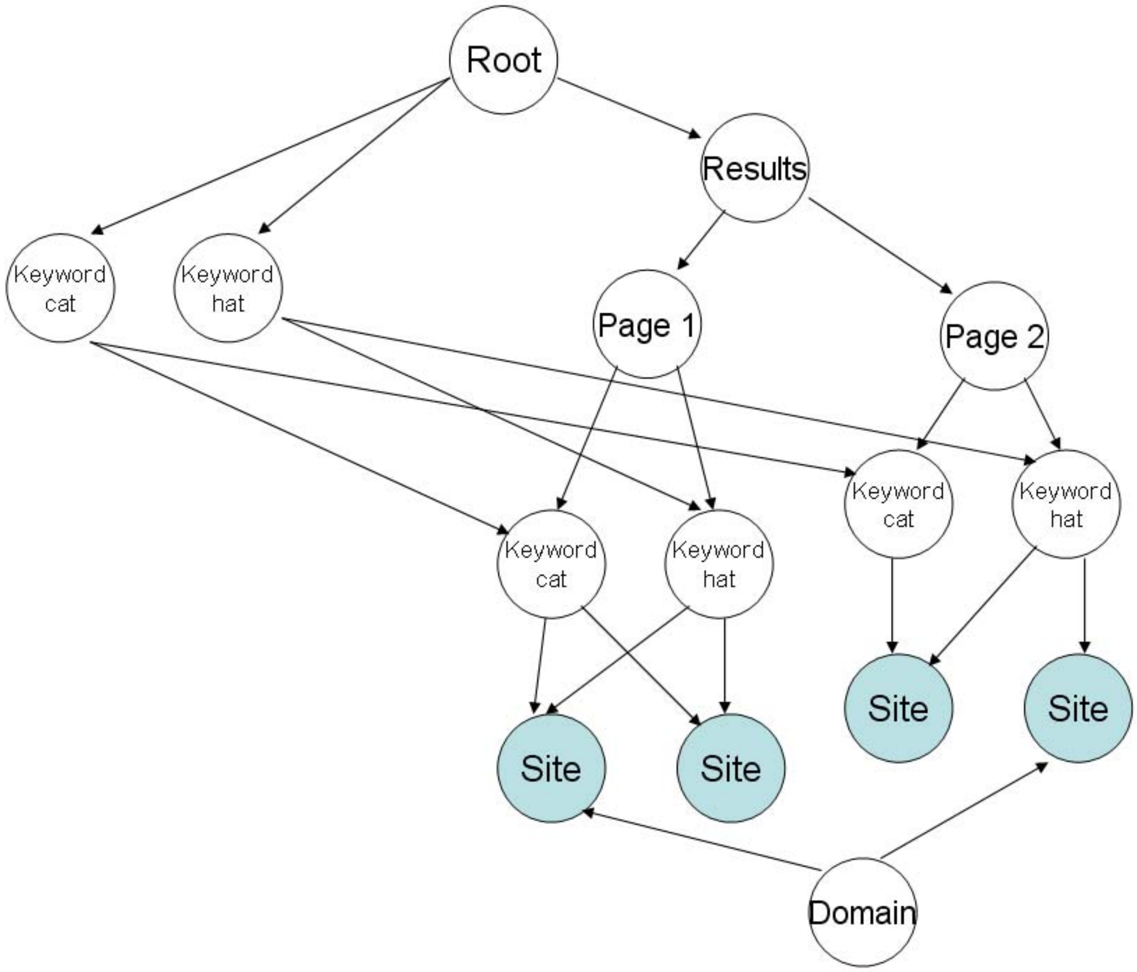


Figure 14. Example format of the created networks

The site nodes are represent a single item returned from the Google search and are directly connected to only the keywords of the page they were grouped with. The connections to the keywords are only created by looking at the title and summary from the item returned by Google. So this means that if the summary Google creates or the title of the node does not include any of the searched keywords then the site node could have no parents. To add a little spice to the network a domain nodes will be created when any two or more sites belonged to the same domain for example: www.google.com and code.google.com would both be connected to a domain node for google.com.

By default searching Google with a few keywords (ex: cat hat) will do an AND search and all results must contain the keywords. This was not desirable since all site nodes would always be linked to all keywords. However, using an OR (ex: cat OR hat) didn't work correctly because it didn't seem to care about sites with both keywords. So instead a more complex search query was used that used AND and OR (ex: (cat AND hat) OR cat OR hat). This new query seemed to cause sites with all keywords to be higher on the search returns then those with just one and some testing showed that sites without every keyword still could be returned on the first few pages. To handle the problem where only 60 search results could be gotten using the AJAX API, each keyword would

be searched separately after the combined search getting until all searches were done or the desired node count was reached.

Using the same idea for layer creation that was done with the 2008 EP a layer creation dialog was created to make it easy to generate a network of nodes from a Google search. To see what the dialog looks like see Figure 15. For the search terms section in the dialog this is where all the keywords can be entered separated by spaces. The code will generate the search query using the list of keywords entered; however this generation only works for up to four keywords. Also the node count is just a suggestion and does not guarantee that the created network will have this many nodes in it. This node count is used for the number of search results to try to get from Google and then deciding when to stop placing search result nodes into the network. Because of the domain nodes that are created at the end of the process the node count is normally greater than the number entered in the dialog.

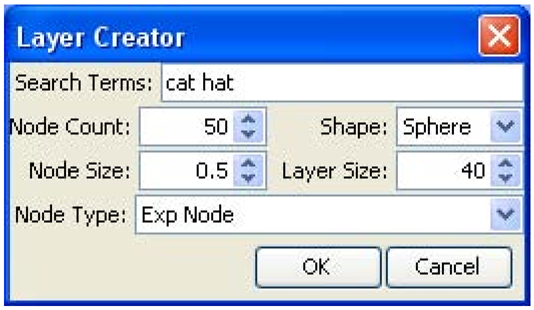


Figure 15. Example Layer Creator dialog

Network Layout

After a network has been created it needs to be laid out in 3D space in some way. The goals of a layout algorithm were:

1. To be able to tell it a volume (in this case a simple box) which all the nodes need to fit into.
2. Having no nodes overlapping each other.
3. Trying to group linked nodes closer to each other.
4. Trying to avoid overlapping of links with other links and nodes not part of the particular link.

In researching for an algorithm it didn't seem like 3D layout of graphs have been tackled as hard as the 2D case. Elisabeth Fitzhugh found a promising algorithm which was described in the paper: <u>Unified Mapping of Social Networks into 3D Space</u> which was written by Jiaqian Zheng and Junyu Niu. This algorithm made use of a 3D construct called an octree. The main thing about an octree is that it takes a space (item 1) and cuts it in half in all three dimensions to create eight smaller sections (an octant) which are all the same size. Each octant can then be cut again into eight smaller sections and repeated to whatever depth or size you want.

The main parts of the algorithm (see the paper listed above) are to map ever node in a graph into its own octant based on its closeness to other nodes. Then the nodes are given the 3D position based on the center of the octant they were placed in when the

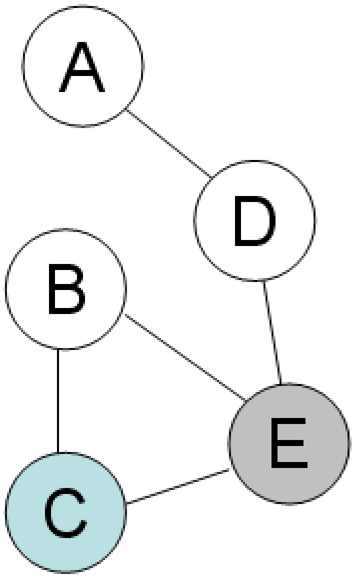
octree is given a box defining its size. The following is a brief description of the mapping of the nodes to octants in the octree based on closeness:

1. All nodes are treated as a root octant.
2. The two root octants with the largest closeness value are grouped together creating one less root octants.
    - Closeness is computed as a leaf octant to leaf octant comparison.
    - If an octant is not a leaf (has children) then the closeness commutation will be done computed as the average closeness computed between all its children to the other octant.
3. Step 2 is repeated until there is only one root octant.

The implementation of the algorithm could not be done exactly because the part algorithm for computing closeness didn't make sense. The paper showed this value being computed by taking the cosine of two vectors (at least that is what it looked like in the paper) which doesn't seem useful if even possible. This operation was still clear as mud after getting a response from one of the authors. It was decided to use the dot product of the two vectors instead of cosine because the dot product definition does include cosine in it, but mostly because the paper stated that a larger closeness value means the item is closer. Since the two vectors (one represents each node which closeness is being computed for) are rows from a matrix created from the network (so given that there are N nodes in the network the matrix (M) is NxN with the weight of the link between nodes i and j being the value at position Mij and Mji, and a weight of zero is used in the case of no link) by setting the diagonal of the matrix (Mii where i is 0 to N) to be the same as the weight between two nodes linked together (all nodes in my generated networks have the same weight) it gives the results:

- For nodes that are linked together the closeness is not zero and double the value of a shared link (item 3)
- The closeness value will be bigger for every node both nodes link to.
- The closeness value will not be bigger for nodes only one of the two nodes links to.

See Figure 16 for an example of a simple matrix of a graph and dot product of two nodes. In this figure the green numbers of the dot product are caused by the two nodes linking to each other.



	A	B	C	D	E
A	1	0	0	1	0
B	0	1	1	0	1
C	0	1	1	0	1
D	1	0	0	1	1
E	0	1	1	1	1

$$C \bullet E = (0 \times 0) + (1 \times 1) + (1 \times 1) + (0 \times 1) + (1 \times 1)$$
$$= 0 + 1 + 1 + 0 + 1 = 3$$

Figure 16. Sample dot product used for closeness

After implementation and testing the dot product version of the algorithm a problem was encountered with clumping of the nodes (violates item 2) as seen in Figure 17. What is happening is that the layout algorithm will keep going deeper in the octree (reducing the size of the octant) without considering the size of the node that will be placed in the octant. So the octree algorithm was tweaked a little to help to improve its layout. The main thing that was changed was giving the algorithm a maximum depth that an octant could be created at which was computed from the size of the nodes and the size of an octant at a given depth so that the node will fit without going into another octant's space at the same depth. In the mapping part of the algorithm when the two closest octants are found the original algorithm was followed if it will not violate the maximum depth rule. If the depth rule would be violated in the normal way then it was attempted to group the two octants together by adding one as a type of child to the other. The first try attempt just fit the node that can be a child into the parent as a child, grandchild, great-grandchild etc. If this way would not work a more complex add was tried, which could take node to become the new parent of a parent node's child where it takes the place of the child, or adding a new node in the parent which has the node and one of the parent's child as its first children. In all cases where a node changes depth it is assured that the maximum depth rule will not be violated for a node or any of its children of any depth. Some randomness was also added to the layout algorithm by randomly placing the node somewhere in their octant so that it is still fully contained, but no longer always in the absolute center of it. This helps a little to avoid a grid type feel for the laid out nodes. The

algorithm still tends to clump most of the nodes in the same area, but know no two nodes ever overlap each other.

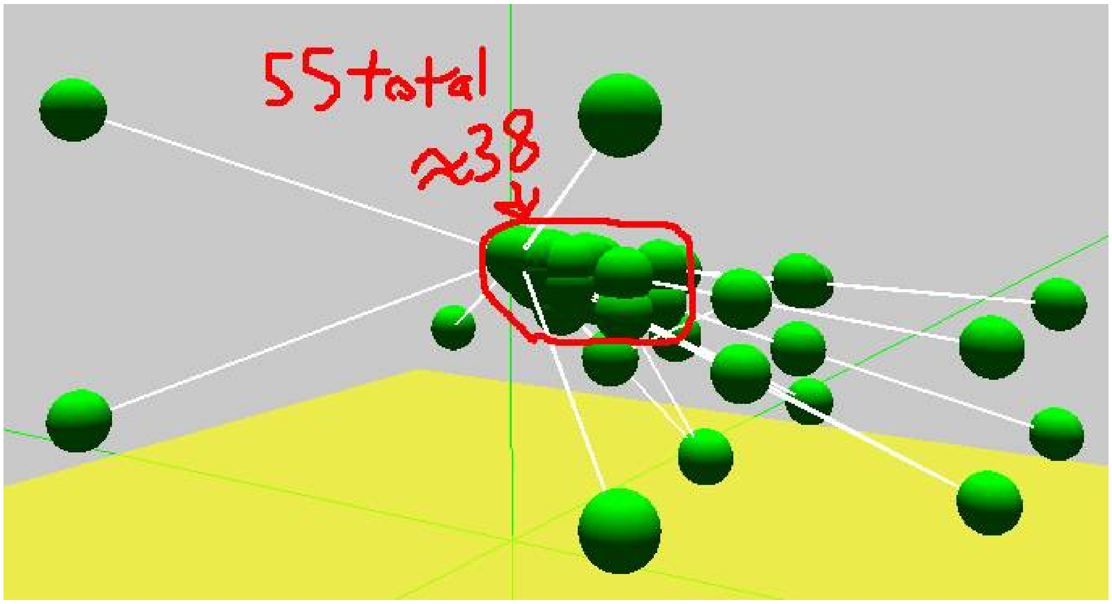


Figure 17. Clumping problem with octree layout

Out of the four layout requirements the modified octree layout algorithm assures that items 1 and two are both satisfied. Item 3 should also be handled pretty well because of the closeness calculation part of the algorithm. Item 4 is being violated, but it was just a request more than anything else. When you have so many links as the networks being created from a Google search and being limited to a finite space (in this case one that is fairly small) it is hard to avoid nodes being overlapped by links.

Target Node Selection

Before the target nodes could be selected for the actual experiment the networks had to be generated so there would be nodes to pick. Since the experiment design called for three network configurations for each network size a total of six networks from three different set of Google search keywords (the same keywords where used to create small and large networks) were created. These are the keywords used to create the networks:

- green eggs ham
- one fish two
- red fish blue

The keywords used came from titles of Dr. Seuss books (minus the common words that Google removes from its searches). The small version of the network was created with around 100 nodes each while the large were created with around 300 nodes.

When picking the target nodes there were a few guide lines which were needed to help provide a balanced experiment. The most important guideline was that for all the target nodes both their drop line and drop shadows could be distinguished in the view and were not ambiguous. Another guide line was to distribute the target node selections to four quadrants (Upper left, lower left, upper right, lower right). The upper and lower halves of the screen were based on the center of the target nodes Y position. Any node

that had a Y value over 9 units was “upper” and if it was under 9 units it is “lower”. Using this methodology made the task easier to complete because of the way the vanishing point works in 3D graphics (as well as real life). For the left, right sides of the screen the Z position of the center was used of node for the 90 degree view (because the Z axis is the same as the center of the screen for this view.) On the 45 degree view the center of the screen was used since it was easy to estimate by using the closest and farthest corners of the floor.

The last guide line that was followed was that a target node selected in one view didn’t have to be used in the same way or at all for the other view. This made the task a lot easier, because it becomes much harder to pick a node that works well in both views. The experiment design required that the picking of one target node for lane B, three for lane C, four for lane D, three for lane E, and one for lane F for small and large nodes for each of the view points. So this gave me a total of 48 target nodes to pick overall.

The target node picking was broken up into four groups of 12 nodes each. One for the small networks at 90 degrees, one for the small networks at 45 degrees, one for the large networks at 90 degrees, and the last for the large networks at 45 degrees view point. All four of these groups were treated separately and independent of each other, since it didn’t seem like a subject would ever notice and make use of any pattern between the four groups.

Inside one of each defined groups there was 3 networks configurations, 12 different target nodes spread across 3 question lanes, and 4 quadrants to work with. Also the 12 target nodes were divided into 4 target nodes for each of the question lanes. It was decided to try to distribute the network configurations as evenly as they could across the data that was provided. So each network configuration provides a single target node in each of the four quadrants in a given group. Also each question lane contains a question for at least one target node from each of the network configurations and the two true target nodes for a question lane are never from the same network configuration.

The table, shown below, shows how the four groups have had their 12 target nodes distributed across the network configurations as well as quadrants. As you can see some question lanes or target nodes seem to use the same quadrants in an “unbalanced” way. This “unbalanced” distribution is required given the guidelines and layout algorithm. The layout tends to clump nodes in a corner of the floor (the closest corner on the right side for 90 degrees and the closest corner in the center for 45 degrees) which makes it hard to see and follow the drop lines and shadows for those in the clump. The clumps in the large networks also tend to obscure nodes behind them. In the 45 degree viewing angle the nodes in lanes B and C tend to be on the left side while the nodes in lanes E and F tend to be on the right side (based on the majority of the lane being on that side of the screen.) All this together can result in no usable nodes in a specific lane for one or more quadrants of a network configuration.

Q Lane	TN Lane	Small 90	Small 45	Large 90	Large 45
C B		1 - UR	1 - UL	3 - LR	1 - LL
C C		1 - LL	2 - LL	1 - LR	3 - LL
C C		2 - LR	3 - UR	2 - UL	2 - UL
C D		3 - UL	3 - LL	3 - UL	3 - LR
D	**C**	2 - UL	1 - LL	3 - LL	1 - LR
D	**D**	1 - UL	2 - UL	1 - UR	2 - LL

D	**D**	3 - LR	3 - LR	2 - LR	3 - UL
D	**E**	3 - LL	1 - LR	2 - UR	2 - UR
E D		2 - LL	3 - UL	1 - UL	1 - UL
E E		2 - UR	1 - UR	3 - UR	1 - UR
E E		3 - UR	2 - LR	1 - LL	2 - LR
E F		1 - LR	2 - UR	2 - LL	3 - UR

Key: *The #-# can be thought of as [Network Configuration Index]-[Quadrant]. So for example "2 – LR" means that the target node is from the second network configuration and is in the "lower right" quadrant.*

Using the Experiment Platform

In the following section two specific versions of the Experiment Platform (EP) are referred to which are used together to create, test, and analyze the experiment. The main version of the EP is the testing software which runs in full screen mode and will run a subject through all the trials (this will be called the testing version). This is for testing subjects using an experiment movie file where the trials will be shown in a random order. Figure 18 shows what the startup screen looks like for the testing version (after it has entered full screen mode). If this version is run with a command line argument of "no_random" the order of the trials will always run in the same order they appear in the experiment movie file and not in random order. This is useful for testing an experiment movie file to make sure it creates the correct output, but should not be used for testing subjects. The second version of the EP is an Eclipse based windows program such as the 2008 EP and is used in experiment creation, and creating Excel data (this may also be given the name of the design version). Figure 19 shows what the startup screen looks like for the Eclipse version.

Figure 18. Initial screen for testing EP after entering full screen mode

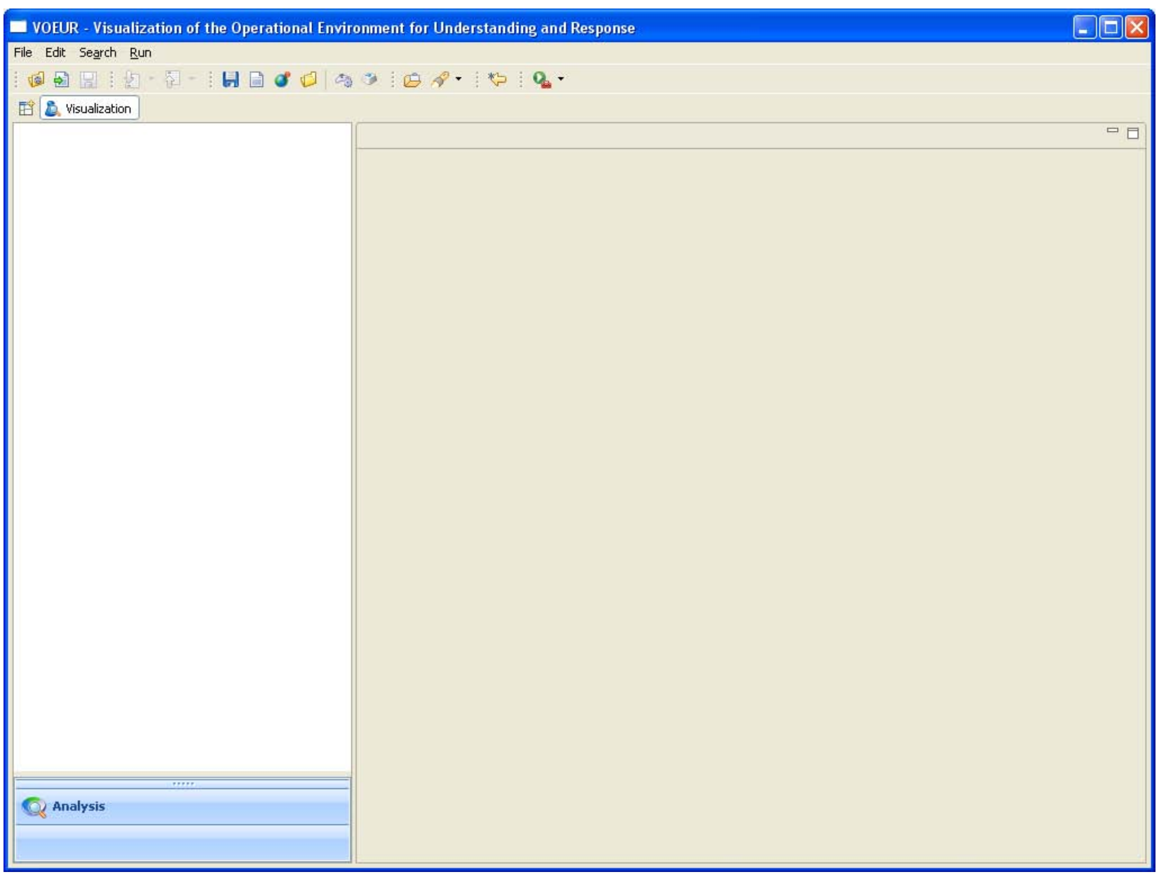


Figure 19. Start screen for the Eclipse version of the EP

Testing Subjects

Configuring the Gamepad

Configuring the gamepad is needed because it tells the program, which of the buttons on the gamepad are to be mapped to the actions the EP understands. This should already have been done for you and the following steps should only be followed if the gamepad is not working correctly or a different gamepad needs to be used.

To configure the gamepad you need the following things:

- Main computer
- Gamepad
    - Connected to the main computer
- Main folder
    - This is where all the other files and folders that are needed will be located as well as being the working directory for running the program
    - The main folder is C:\SRAProjects\Experiment3-18 (The build date 3-18 could be a later date at the time of the session.)
- Design EP
    - The design EP is located in the design EP folder of the main folder.

The following steps explain how to configure the gamepad.

1. Run the ep.exe program located in the design EP folder, after the gamepad has already been connected to the computer.
2. Click the toolbar item for configuring the input device as shown in Figure 20.
3. After clicking the toolbar item the Input Binding Dialog should appear as shown in Figure 21.
4. Just to be safe click the Remove All Bindings button first and then click the “Yes” button when you are shown the dialog in Figure 22.
5. Now make sure that the “Input Device” combo box is selected to the gamepad you want to configure and make sure it is enabled. It is also best to also make sure that the keyboard device is disabled.
6. While on your desired gamepad input device select “All” and “Key Actions” from the “Specific View” combo boxes.
7. Next select the action you want to setup a binding for (This needs to be done for SELECT, LEFT, RIGHT, and BACK) by clicking on the button. Figure 23 shows the result of clicking the SELECT button.
8. You next need to select the correct button to map to from the “Input Component” combo box. To find out what a specific button’s name is you can press that button on the gamepad and the text area will display a log of all the input events that have been collected (shown in Figure 23). By pressing the button a few times you can verify the name of the button you are pressing. *Note: analog sticks can also be mapped to an action, but it is much more completed to do, so I will not go into it here since it shouldn’t be needed.*
9. After you have selected the correct button to map to press the “OK” button in the “Control Binding” group. This is not the main “OK” button shown in the lower right corner of the dialog.

10. When all the bindings have been setup press the main “OK” button in the lower right corner of the dialog to close it.
11. The steps so far are only for changing the key mappings for the Design EP and not the Main EP used for running the experiment. So you must copy over the updated mappings file to the location where the Main EP will use it.
    a. Copy the keys.properties file in the “design EP\properties\voeur” folder of the main folder.
    b. Paste and replace the keys.properties file which is located in the “voeur” folder of the main folder.

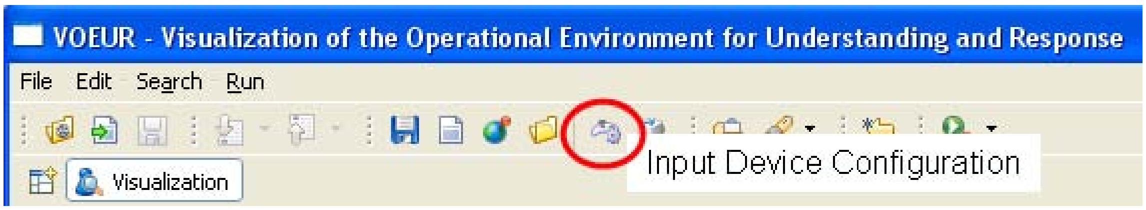


Figure 20. Button to press to open the Input Binding Dialog

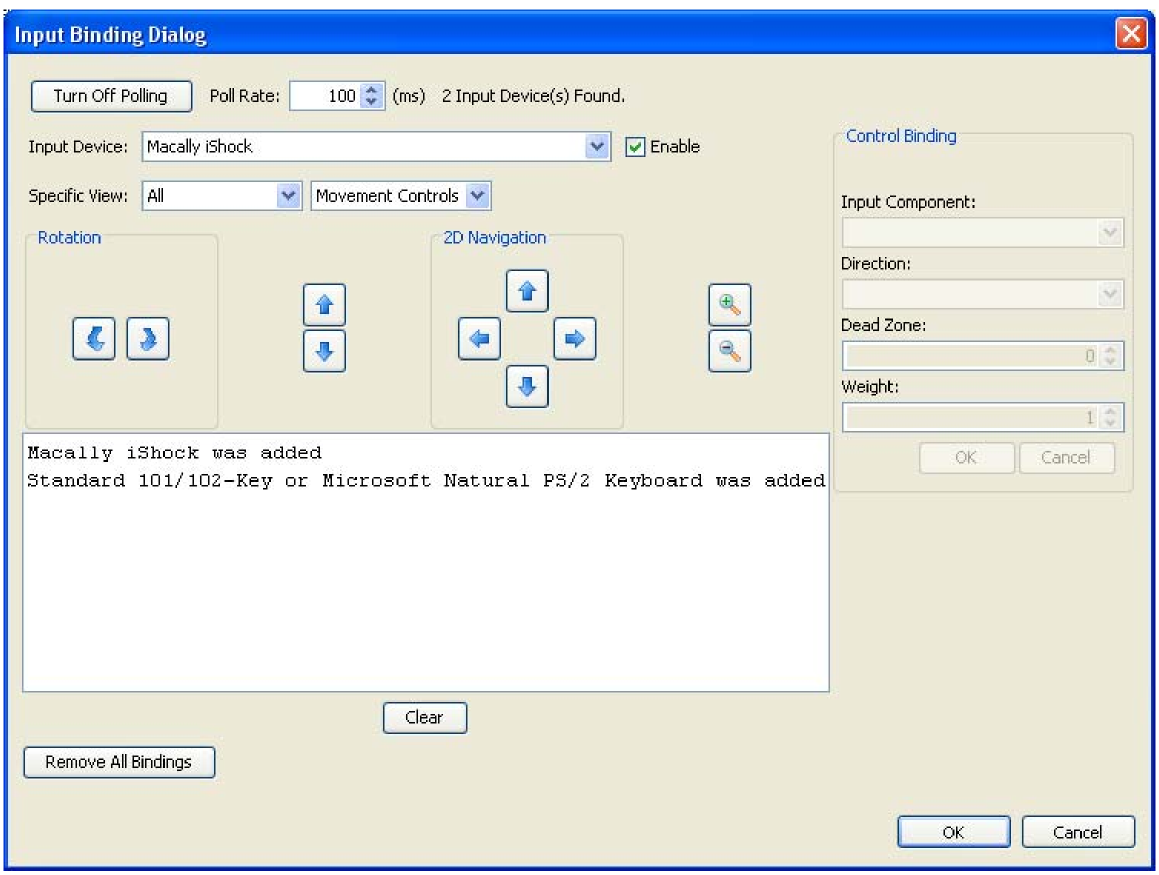


Figure 21. Input Binding dialog

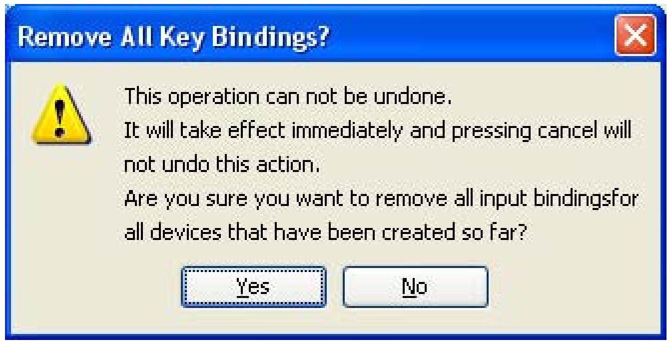


Figure 22. Are you sure you want to remove all the key bindings?

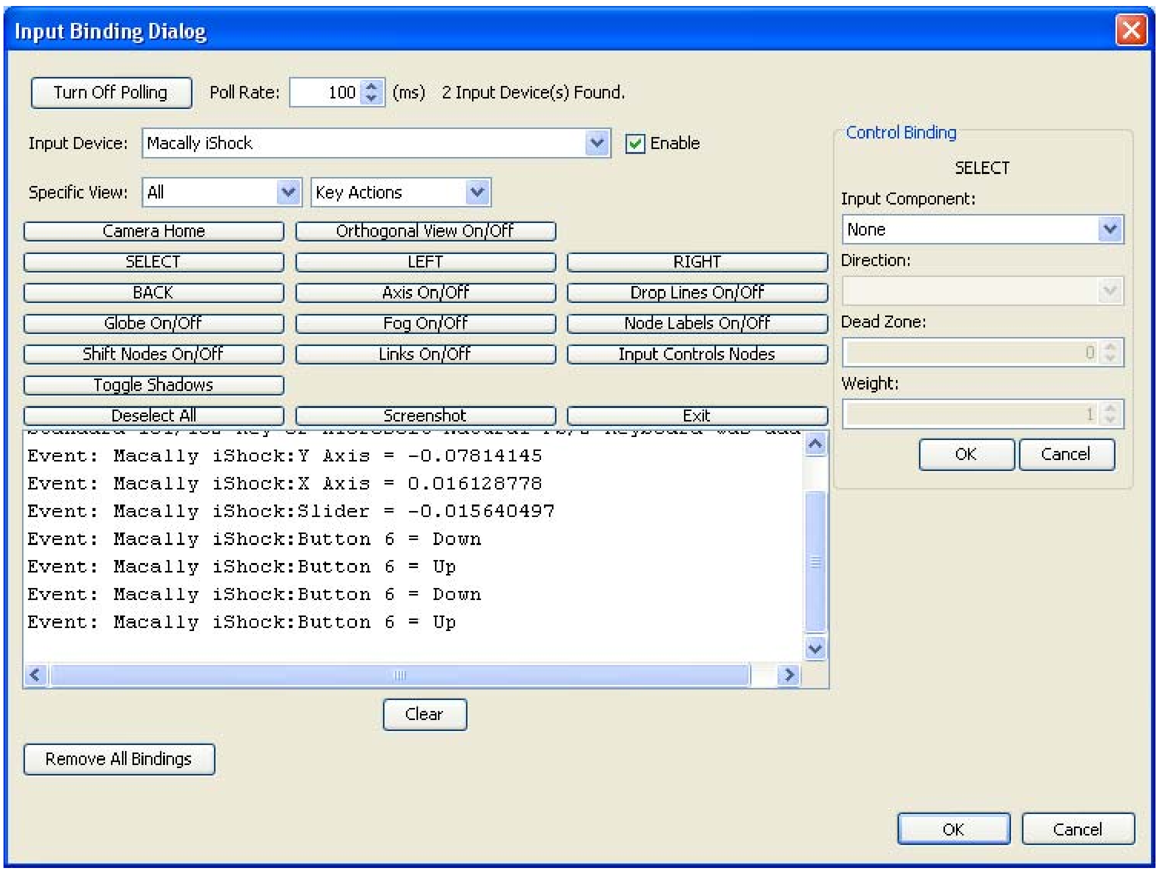


Figure 23. Key actions

Running Subjects through the any Session

Since the same process is followed for any session including 2.5D, 3D as well as training, this section handles it all by adding notes for handing the different session types. To test subjects you need the following things:

- Main computer
- Gamepad
  - o Connected to the main computer

    - The gamepad must have already have been configured (See the **Configuring the Gamepad** section for more information)
- Shutter glasses (**3D session**)
- Main folder
    - This is where all the other files and folders that are needed will be located as well as being the working directory for running the program
    - The main folder is C:\SRAProjects\Experiment3-18 (The build date 3-18 could be a later date at the time of the session.)
- Batch file
    - experiment.bat
- Jars
    - ep.jar
    - The jars folder
        1. This folder contains other required jar files needed, like jview.jar.
- VOEUR files
    - phaseOne.owl
    - main.properties
        1. Located in the folder phaseOne.owl.data
- Session files
    - phaseOne.txt
    - phaseOneTrain.txt
    - phaseOne_25D_1.txt
    - phaseOne_25D_2.txt
    - phaseOne_25D_T.txt
    - phaseOne_3D_1.txt
    - phaseOne_3D_2.txt
    - phaseOne_3D_T.txt
- Other folders
    - images
    - voeur
        1. *Note: This is where the gamepad configuration is stored.*
- Other files
    - cogviz.properties
    - jinput-dx8.dll
    - jinput-raw.dll
    - jinput-wintab.dll
    - nodeMapping.properties

The following steps are for running a subject through any session.

1. The first thing you need to do on the main computer is to turn off the screen saver so that it does not turn on during the session. This is because no input or mouse movement will be done on the main computer during the session.
2. Run the batch file experiment.bat located in the main folder.
3. Before the experiment file begins to load, the EP will ask for the subject's ID. This dialog is shown in Figure 24. This should be the same value used for the subject going through both sessions.

4. After the pressing the “OK” button on the subject ID dialog a dialog will ask you what display mode you want to use (Figure 25).
    a. For **2.5D sessions** press the “2.5D” button.
    b. For **3D sessions** press the “3D” button.
5. Once the display mode has been selected the last setup dialog will appear (Figure 26) asking you what session the program should be setup for.
    a. If the subject is going through a **training session** press the “Train” button.
    b. For the subjects **first time through a session** for the previously selected display mode in the actual experiment (not training) press the “1” button.
    c. For the subjects **second time (repeating) through a session** for the previously selected display mode in the actual experiment (not training) press the “2” button.
6. After the desired session has been chosen the experiment movie file will load and the ontology file that goes along with the experiment movie file will be loaded as well.
7. When loading has finished the EP will go into full screen mode and will provide an experiment ready screen that looks something like that shown in Figure 27.
8. (For **3D sessions** only) When the program goes to full screen mode it will have started working in true 3D mode although it will not show anything in true 3D until the first trial is shown to the subject. At this point you need to make sure that the shutter glasses system is currently active. The little black NVIDIA box thing should have its label on the front lit up when it is on. Also make sure the shutter glasses are on as well (there is a button on the left arm of the frame.)
    a. If the NVIDIA box thing is not lit up try pushing the button (the label) and pushing button on the shutter glasses.
    b. If the box thing is still not lit up at this point try to restart the computer.
    c. After a restart if the NVIDIA box thing still does not light up when the program is running and the button has been pressed seek help from IT.
9. When you have pressed the SELECT button on the gamepad (the button mapped to the SELECT action and not necessarily a button labeled “Select”) the final setup progress will start (this can take a few seconds to complete) before the first ready screen is shown. There is a problem with the shadows that sometimes the program fails to get the context to the hardware buffer. So to fix this at this point in time the program tries to render shadows and will automatically quit the program if it fails (which has never been seen after this fix was put in place.) So if the program quits at this point unexpectedly return to step 2. *Note: At this point the experiment timer which times how long it takes to compute the experiment will start*.
10. The first user ready screen, similar to the one in Figure 28, will be shown. The subject needs to be setup to run the session by this point.
11. When the subject clicks the SELECT button the first trial question will be shown.
12. The subject will then go through the process:
    a. Viewing the trial image.
    b. Pressing the SELECT button to notify the computer they are ready to answer the question.

c. Selecting the yes/no part of their response to the trial question in displayed menu. (see Figure 29)
d. Selecting the confidence level to finish their response to the trial question. (see Figure 30)
e. After the response was been given the next ready screen will be shown. (See Figure 31)
f. Pressing the SELECT button will start showing the next trial image.

13. When the subject gets through the session, which will probably take one hour, (this depends on how fast they are and how much time they spend on breaks) after finishing all the trial questions in the session they will see a screen similar to the one in Figure 31. The mouse cursor which has been hidden during the experiment process will now be visible again at this point. At this point the subject has finished the sessions and their Excel data has been written to their user movie file. *Note: by pressing SELECT one more time will result in a completely blank screen. There is no difference between states from the final screen in Figure 32 and this blank screen. So the program can be safely shutdown at either of these points.*
14. The user movie file that is created will be located in the main folder with the name along the lines of usermovie*_ID_.txt where * specifies the display mode and session number value and ID is the value entered in for the subjects ID. So if you entered in 5 for the subjects first 3D session it would be: usermovie_3D_1_5_.txt. If a file named usermovie_3D_1_5_.txt already existed in the folder and you used subject ID of 5 again you would get a filename of usermovie_3D_1_5_001.txt. If the same subject ID is used more times then the 001 value will be incremented so that every user movie file has a unique filename even if the same subject ID is used.
15. When the subject is finished with the experiment you will need to quit the testing EP and the input program. To quit the testing EP (make sure the program has keyboard focus by clicking the left mouse button) just click the F5 button on the keyboard.

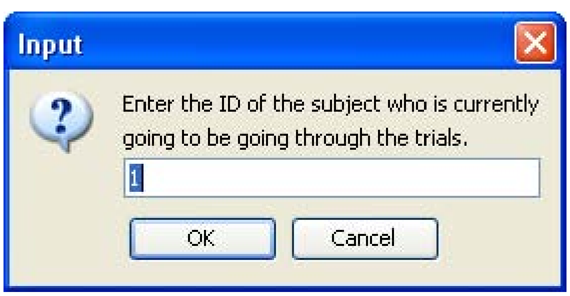


Figure 24. The first dialog, which asks for a subject ID

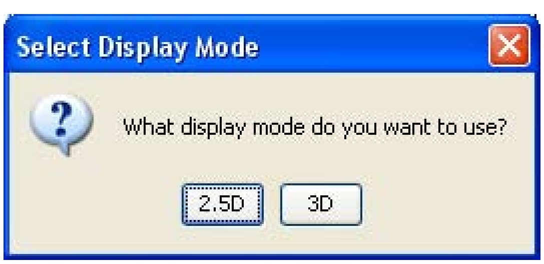


Figure 25. The second dialog, which asks for the desired display mode

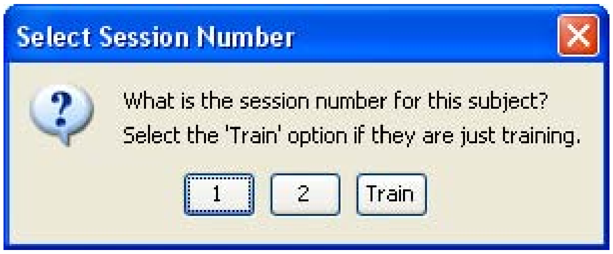


Figure 26. The third (and last) dialog, which asks for the desired session

Figure 27. Experiment Ready screen

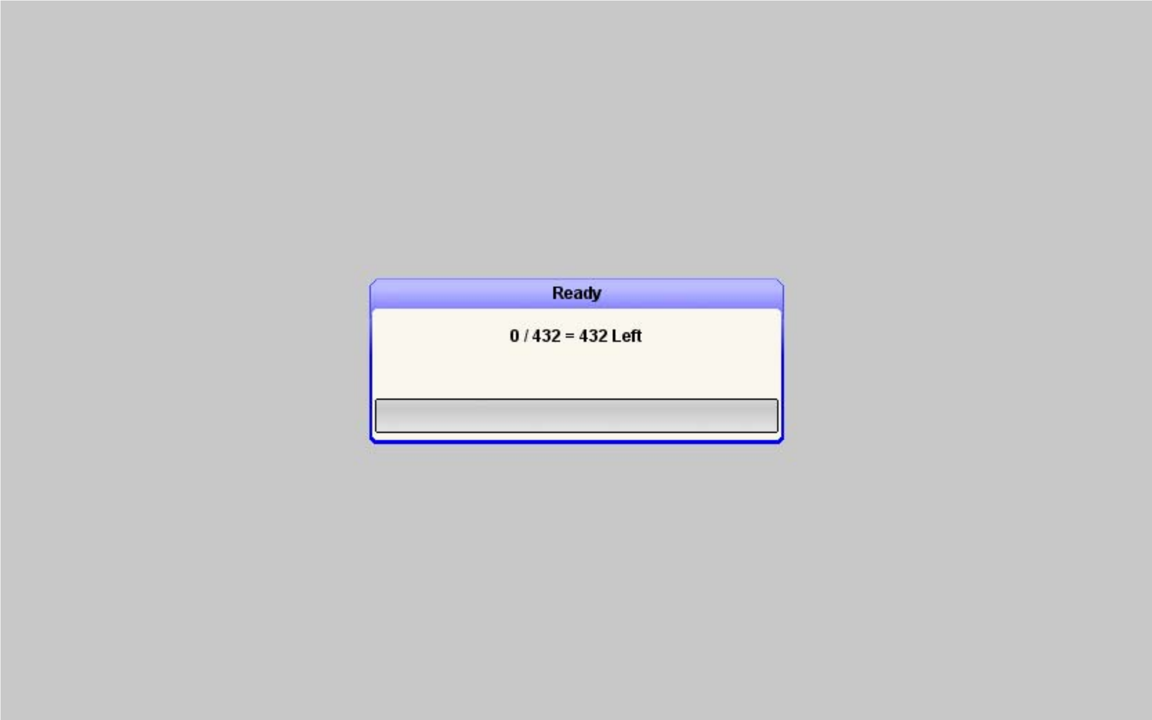


Figure 28. Sample of the first User Ready screen

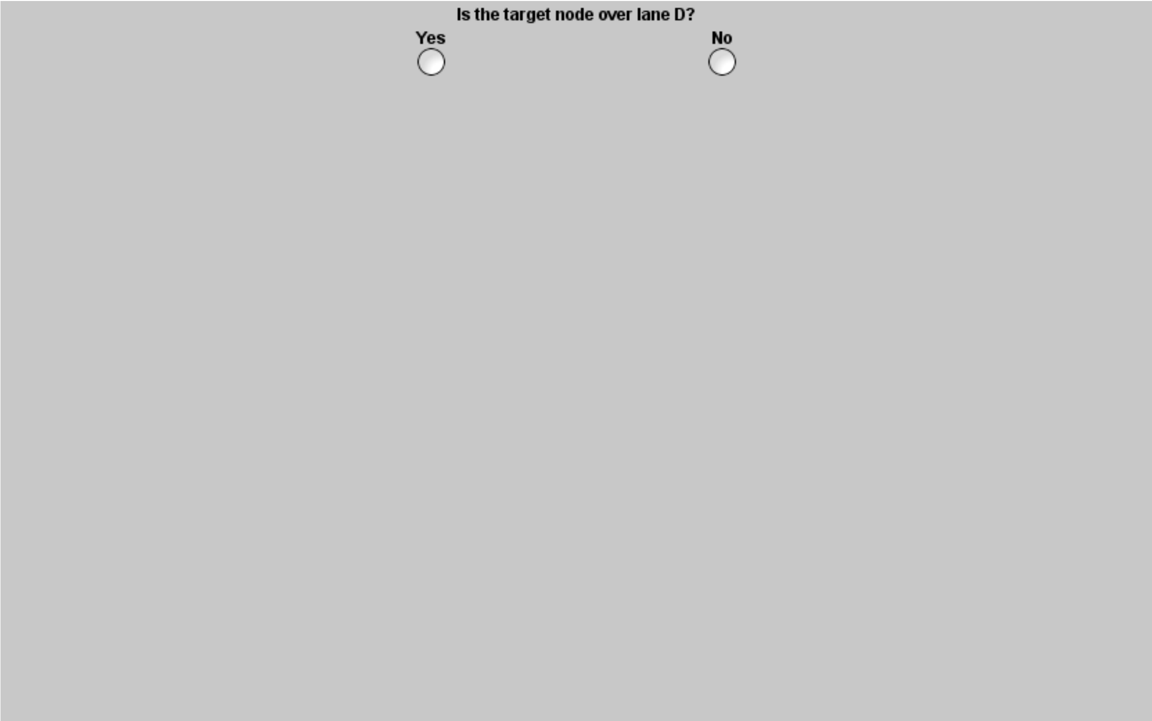


Figure 29. Response "menu"

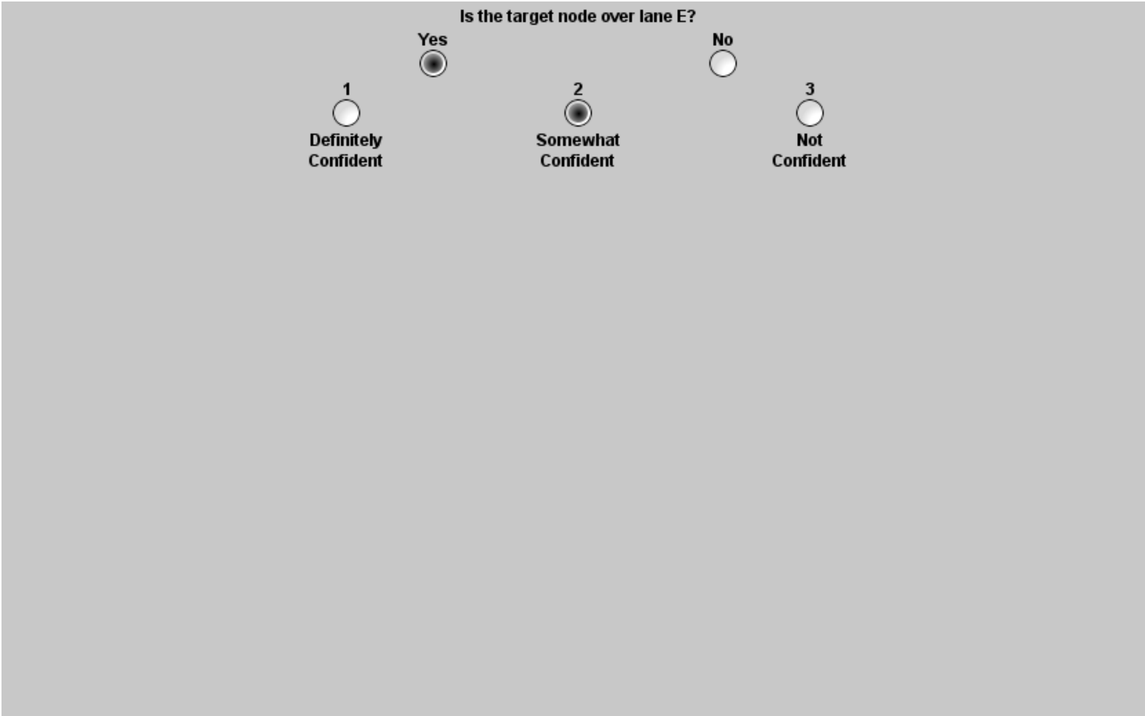


Figure 30. Confidence level "menu"

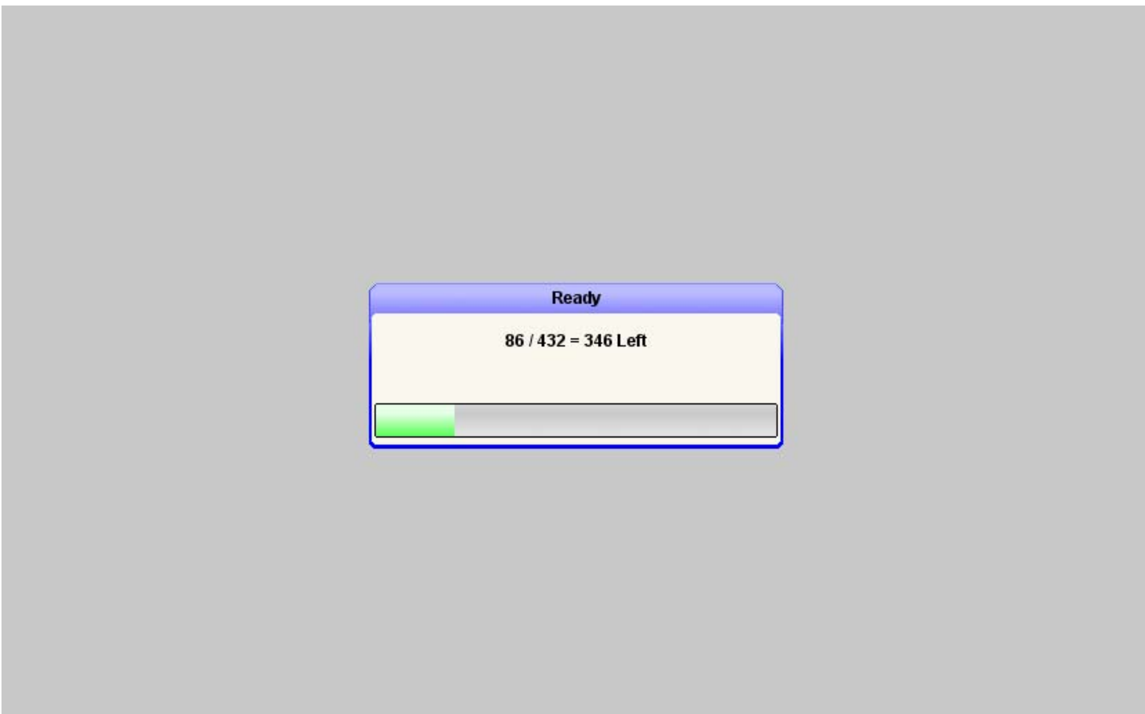


Figure 31. Example of a Next Ready screen

Figure 32. End of Session screen, signaling end of session

Other Subject Testing Info

When the dialog box shown in Figure 24 appears asking for the subject's ID, it is the administrator's job to make sure the subject has the same ID for both sessions they complete. Also the ID should be unique to the subject, so no two people should have the same subject ID. The subject ID is shown in the first column of the outputted Excel data which will be discussed later. The dialog will accept almost any string so letters and number can be used. For the subject's privacy numbers or just a letter should be used for the subject ID such as (1, 2, 3, and 4 or A, B, C, and D.) After the subject's id dialog box will be followed by a dialog box for selecting the display mode to use (2.5D/3D see Figure 25), and then one for selecting the session number (See Figure 26). If any of these three dialogs are closed without selecting a valid option (such as pressing the "X" in the upper right corner) the program will quit. Loading of the experiment movie file and corresponding owl files will begin after the session has been picked. When full screen mode has been entered (see Figure 27) and the SELECT has been pressed, it will take a few more seconds to startup the experiment. Just a gray background will be shown during the experiment startup process, and after everything has finished the screen will show the first user ready screen (see Figure 28) in the center of it.

When you are on the initial ready screen (see Figure 27) this is a good time to sit the subject down and get them ready to start the experiment, but you can sit them down at any time before this point if you want. There is a timer that times how long it takes the subject to complete the experiment in the movie file (This is different from the total time it takes them to answer the questions because it includes time spent on the in between dialogs for the session.) The timer and experiment will not start until the box in Figure 27 has been dismissed. When the testing EP shows a box similar to the one in Figure 32 (the

end of experiment session box) the subject has finished going through the session of the experiment. Also at this point the Excel data for the user has been written out to the user movie file which contains the actions that the subjects did for the whole experiment. If the subject or administrator quits the EP before the end of the experiment session box has been shown, Excel data will not be written out to their user movie file allowing them to drop out of the experiment without having their data used. When you reach the experiment session box the administrator can close down the EP (by pressing the F5 key on the keyboard when the program has the keyboard focus. To make sure the program has the keyboard focus you can click the left mouse button once). The EP can only handle a single session so it must be shutdown and a new instance started to run another session/subject.

When running through one of the session files for the experiment between each trial the subject will be shown a next ready screen. An example of the next ready screen is shown in Figure 31. The next trial is not shown until the subject presses the SELECT button. The ready screen is used as a pause mechanic allowing the subject to take a short or long break if they need it while going through the experiment. However, it should be noted that the timer that keeps track of how long it takes the subject to get through the experiment will include idle time spent on a dialog like this. So if a subject takes a five minute break half way through the experiment that five minutes will be included in the total time it took them to complete the experiment. As you can see in Figure 24 the ready screen gives feedback to the subject on their progress through the experiment. This way they know how many questions they have left and how far they are through the experiment session as a whole. The subject will then have five seconds to view the 3D scene before it times out if they have not stated they are ready to answer the question (stating this also causes the 3D scene to disappear). After the subject has stated to the computer they are ready to answer the question for the 3D scene currently being displayed (done so by pressing the SELECT button) they can take as much time as the need to enter their answer on the answer menu. While the subject is on the answering menu there will be a timer running to track how long it takes them to submit their answer. So subjects should only take a break while they are on the ready screen (the ready screen is shown after trial).

During the time that the subject is going though an experiment movie file with the data about their answers and selections is being recorded and stored in a user movie file created by the user movie recorder. This file has two purposes, to store Excel data and recording all of the subject's actions during the experiment which can be reviewed later (the watch feature is currently not available). By default the user movie files will be created in the same directory as the testing version of the EP with the filename following the naming convention: usermovie<session>_<subject id>_###.txt. The <session> part of the filename tells what the display mode is (2.5D/3D) as well as the session number the movie is from. The <subject id> part of the filename displays the ID is the subject id entered in the corresponding dialog. The ### part is in place to make sure that user movie files for the same session and subject are not overwritten by the EP. If no files currently exist with the filename usermovie<session>_<subject id>_.txt then this will be the name of the file created. However, if a file of this name exists then the ### will be added where ### is a number starting with 001 and going up to 999 so that the filename will have a unique name.

Watching Subjects Performance

This feature has not been implemented for this experiment, but the user movie files that are created during a session running a subject should contain everything they need to watch the subject. It would just take a little bit of work to re-implement this into the experiment platform (EP). In the setup and creation of the EP the code from the 2008 EP was never fixed and tested because it didn't seem useful to be able to see what the subject saw and did for this experiment.

Creating Excel Data

If you scroll down to the end of a completed user movie file you will that some the Excel data has been implanted inside the movie file as comments (A line starting with # in both the experiment movie file and user movie file are comments.) An example of the Excel data block in a user movie file can be seen in Figure 33. The data starts and end with a special tags specifying Excel data start and end points. After the start tag is the headers for the columns and then one line which represents a row and a single question. The data is separated by commas and you could copy and paste this data into a different text file, remove the "# " at the start of each line and then open the file with Excel to view the data in the format it was designed to be viewed as. The Excel data in the file needs to be converted into a format Excel can open by using the Eclipse version of the EP. When the Excel tool is used in the EP it will merge all user movie files it finds into a single tab separated file.

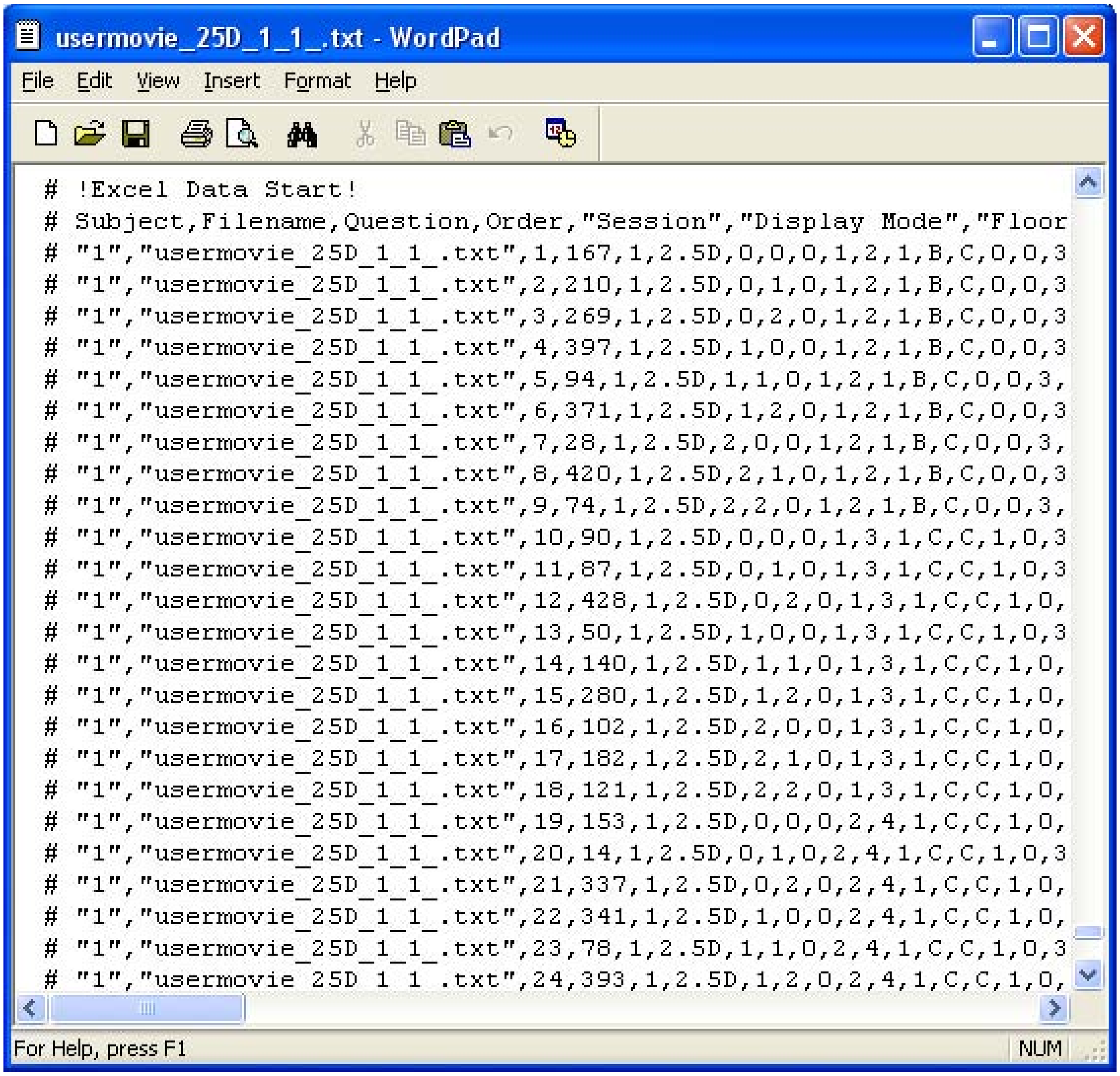


Figure 33. The Excel data placed at the end of a user movie file

One Excel File for Each User Movie File

1. Open up the folder C:\SRAProjects\Experiment3-18\design EP\ (The build date 3-18 could be a later date at the time of the session.) and remove all files that start with the string "usermovie".
2. Next drag and drop (or copy and paste) the user movie file that you that you want to create Excel data for into the C:\SRAProjects\Experiment3-18\design EP\ folder.
3. Now run the Eclipse version of the EP (ep.exe) located in the C:\SRAProjects\Experiment3-18\design EP\ folder if it is not already running. You do not have to close the EP when you are creating Excel data files from user movie file(s) until you are done creating all the Excel data files you want to create.
4. When the EP is open click on the "Merge User Movie Data To An Excel File" toolbar button shown in Figure 34.

5. This will open up a save dialog where the name of the Excel data file to create needs to be entered. The default folder location which files will be saved in will probably start as the user's "My Documents" folder, but this can be changed to a different folder (changing and using that folder should cause the new folder to be the default folder the next time the save dialog opens).
6. Click the "Save" button to create the Excel data file.
7. By default the created Excel data file is given a "tsv" extension meaning Tab Separated Values. This file type is not associated with Excel by default so to open it you need to right click the file and select "Excel" from the "Open With" submenu. If you open up the file with Notepad or WordPad it will be very hard to ready the contents of the file.
8. Once the created tsv file is opened with Excel you can "Save As" to create an "xls" file which is Excel's standard format.
9. At this point the user movie file's data is in a state that a program like SAS can probably handle.
10. Before you are done make sure you move the user movie file (If you dragged and dropped it) back where you found it or in a new directory to store handled files. If you copied and pasted the file you can delete the file since the original still exists in the root folder of the EP.

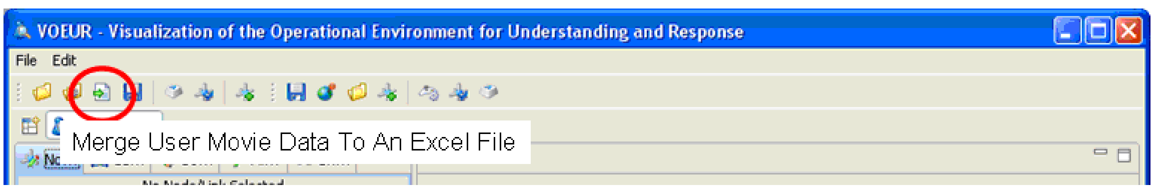


Figure 34. The button to press to merge user movie data into an Excel format

One Excel File for Multiple User Movie Files

Merging multiple user movie files allows the creation of a single Excel file for all the data collected in the experiment including, all session one data, or both sessions for a given subject.

1. Open up the folder C:\SRAProjects\Experiment3-01\design EP\ (The build date 3-18 could be a later date at the time of the session.) and remove all files that start with the string "usermovie".
2. Next drag and drop (or copy and paste) the user movie files that you that you want to merge together to create Excel data for into the C:\SRAProjects\Experiment3-18\design EP\ folder.
3. Now run the Eclipse version of the EP (ep.exe) located in the C:\SRAProjects\Experiment3-18\design EP\ folder if it is not already running. You do not have to close the EP when you are creating Excel data files from user movie file(s) until you are done creating all the Excel data files you want to create.
4. When EP is open click on the "Merge User Movie Data To An Excel File" toolbar button shown in Figure 26.

5. This will open up a save dialog where the name of the Excel data file to create needs to be entered. The default folder location which files will be saved in will probably start as the user's "My Documents" folder, but this can be changed to a different folder (changing and using that folder should cause the new folder to be the default folder the next time the save dialog opens).
6. Click the "Save" button to create the Excel data file. If you are merging a lot of large (Over a megabyte) user movie files together at once there might be a delay before the merging is done. However, my testing with two large files seems to have no noticeable delay after the "Save" button is pressed and the output file is created.
7. By default the created Excel data file is given a "tsv" extension meaning Tab Separated Values. This file type is not associated with Excel by default so to open it you need to right click the file and select "Excel" from the "Open With" submenu. If you open up the file with Notepad or WordPad it will be very hard to ready the contents of the file.
8. Once the created tsv file is opened with Excel you can "Save As" to create an "xls" file which is Excel's standard format.
9. At this point the user movie file's data is in a state that a program like SAS can probably handle.
10. Before you are done make sure you move the user movie files (If you dragged and dropped them) back where you found them or in a new directory to store handled files. If you copied and pasted the file you can delete the file since the original still exists in the root folder of the EP.

Some more things about the how the merge tool works should be noted. The merge tool will search using a base filename, which is defaulted to usermovie.txt. Any file that it finds that starts with this base filename without its extension will be included in the merged file. So usermovie.txt, usermovie001.txt and usermovieHello.txt will all be included while user001.txt would not. The files are merged based on the alphabetical order of the files names so usermovie.txt comes before usermovie001.txt which comes before usermovie102.txt. If a user movie file does not have Excel data embedded inside it the file will be excluded from the merge since it doesn't have anything to add. What this means is that if a subject stops a session early and drops out of the experiment then that session file will not be included in the merge. However, if the same subject already had completed a session the completed session would be included in the merge.

Understanding the Excel Data

The experiment platform (EP) uses Excel data in the same way as the 2008 EP. However, this EP and the 2008 EP do have some different columns. Both utilize the experiment movie files ability to add specific columns to the Excel data that is stored inside the outputted user movie file. Although since the main question is different between the two EP's basic columns which are related mostly to the question and answers have changed. The following shows the basic columns (in black) and the specific columns (E. through N.) of data included in the Excel files from running subjects in phase one of this experiment:

A. Subject

    a. This column stores the subject ID that was entered after the experiment movie file was finished loading. Figure 23 shows the dialog used for setting this value. This should be a unique value given to the person who is the subject so that all sessions they complete will have the same value for this column.
    b. This value is meant to be any integer value, but can be a string value as well.

B. Filename
    a. This tells the file that the given line of data came from. So you can tell which file this line of data came from incase you want a closer look at it.
    b. This is a string that contains the original name of the file that was created. If you renamed the file before merging the original name would be displayed here instead of the renamed value.

C. Question
    a. This column tells the number of the question for the given experiment movie file.
    b. This is an integer that ranges from 1 to the number of trials in the session.

D. Order
    a. This column tells the order at which the subject saw the question connected to this row for the first time.
    b. A value of 1 means it was the first question the subject saw, 100 means it was the 100th question they saw and so on.

E. Session
    a. This field tells which session from a specific display mode this is.
    b. Each session has its own set of question numbers so if you have a merged file with both sessions' data there will be duplicate question numbers, but they will have different session values.
    c. 0 = Training, 1 = first session, 2 = repeated session.

F. Display Mode
    a. This field tells which of the display modes such as 2.5D or 3D were used.
    b. The values are 2.5D and 3D.

G. Floor
    a. This field tells if the type of floor that was shown.
    b. 0 = Grid, 1 = Lanes, 2 = Tick marks.

H. Visual Aid
    a. This field tells the type of visual aid shown.
    b. 0 = None, 1 = Drop lines, 2 = Drop shadows.

I. Network Size
    a. This field is to specify the size category of the network which dictates approximately how many nodes are in the network.
    b. 0 = Small, 1 = Large.

J. Network
    a. This field tells the index ID given to the network related to its network size category.
    b. The value of this field really has no use in making any conclusions from.
    c. A one based index.

K. Target Node
   a. This field tells the index of the target node which corresponds to its position in the network based on the current view point.
   b. The value will be 1 = Upper Left (UL), 2 = Upper Right (UR), 3 = Lower Left (LL), 4 = Lower Right (LR).
L. View Point
   a. This field tells the index ID given to the view point related to the target node being viewed.
   b. The values will be 1 = 90 degree angle, 2 = 45 degree angle.
M. Target Node Lane
   a. This field tells the lane that the target node was over.
   b. The value will be a letter: B, C, D, E, or F.
N. Question Lane
   a. This field tells the lane that the question was asked about.
   b. For example the # in the question "Is the target node over lane #?" would be the value of this field.
   c. The value will be a letter: C, D, or E.
O. Correct Ans
   a. This stores the correct answer to the question.
   b. 0 = No, 1 = Yes.
P. Subject Ans
   a. This stores the subject's answer to the question (with the certainty of confidence level of the answer). In the case the subject timed out Timeout will be the value.
   b. 0 = No, 1 = Yes.
Q. Subject Cert
   a. This stores the subject's certainty value that goes along with their answer.
   b. Has a value in the range of [1, 3] where 1 = Definitely Confident, 2 = Somewhat Confident, and 3 = Not Confident.
R. Timeout
   a. This tells if the user timed out on the question or not.
   b. Because timeouts are not allowed this value will always be 0 (= No.)
   c. 0 = No, 1 or higher = Yes.
S. Correct
   a. This tells if the subject got the question correct or not. If the user timed out then this value will be 0 meaning no. If there is no correct answer to the question then this value will always be 1.
   b. 0 = No, 1 = Yes.
T. TSD Correct
   a. This tells which of the four TSD result the subject got.
   b. 1 = Hit, 2 = Correct rejection, 3 = Miss, 4 = False alarm.
U. Score
   a. This tells a score that was computed based on the answer correctness and the certainty the subject was of the answer.

b. This value is a floating point value in the range of [0, 1] where 0 is for using the highest confidence level for an answer that was wrong and 1 is using the highest confidence level for an answer that was correct.
c. If the user timed out on the question and there is normally a rank the rank will have a value of 0.

V. Ready Time (Sec)
  a. This tells how long it took the subject before they indicated they are ready to answer the question.
  b. This field will include all the time spent looking at the stimuli, so if the stimuli timed out the time will probably be a little higher than the timeout value of 5 seconds.

W. Answer Time (Sec)
  a. This tells how long it took the subject to input their response after they said they were ready to respond.
  b. The subject did not see the stimuli during this time period.

X. Total Time (Sec)
  a. This tells how long it took the subject to answer the question. If the subject timed out then this value will be higher than the timeout value.
  b. This value is just Ready Time + Answer Time.

Y. Running Time (Sec)
  a. This tells how long it took the subject to answer the question since the start of the session.
  b. The max value in this column will provide the total amount of time through the experiment, which includes break time.
  c. Sorting by this column will provide the order that the subject answered all the questions which will be different from the Order column because that is for when they first saw the question. The answer order can be different due to timeouts.

Addendum

Experiment Movie Format (September 2008):

The following is the contents from a document that was created for the 2008 depth cue related experiment. The document describes most of the features of the experiment movie file format and how they work to create different types of trials. This format was used as a basis for this year's abstract data sets experiment. One small change in the documentation is that where it says localname for the identification of an object like a layer or node it now needs the URI since the localname is not guaranteed to be unique, but the URI is.

```
-----------------------------------
# This is a comment in the experiment movie file
# Comments and blank lines can appear anywhere in the file

# There are two types of tags lone tags and tags with parameters
[Lone Tag]
[Tag With Param] Param
# Parameters always start one space after the closing ]
# Tags are always of the format where each word has its first letter
```

```
# capitalized and words are separated by spaces. Blocks always have
# a start and end tag which always end with the word start or end
# for example
[Block Start]
[Block End]
# Where the first tag is the start of the block and then second tag
# is the end of the block

# An experiment movie file must begin with the tag
[Experiment Start]
# No other tags can appear before this tag. Comments and blank lines can.

# The end of the file is specified by the tag
[Experiment End]

# All trials to be run must be in between the Start and End tags

# An experiment movie file has to tell what owl file to use
# this is specified with the following tag and parameter
# The param should be an absolute filename of the owl file to use
# This tag must be between the Experiment Start and End tags and
# outside of a trial block
[Experiment File] Parma

# Outside of a trial block you can define up to 10 global variables using the tags:
[Variable1] Param
[Variable2] Param
[Variable3] Param
[Variable4] Param
[Variable5] Param
[Variable6] Param
[Variable7] Param
[Variable8] Param
[Variable9] Param
[Variable0] Param
# Where the Param is what the variables value will be.
# Now for the ten different variables they can be used any where using the
# following strings %{Variable1} %{Variable2} %{Variable3} %{Variable4} %{Variable5}
# %{Variable6} %{Variable7} %{Variable8} %{Variable9} %{Variable0}
# The global variables must be defined first before these string will be replaced.
# When ever a line of the file is read in it will check for the global variable strings
# and perform a substitution of that the variable string with that variables value
# For example you could use a global variable to represent at tag like in the following
# example:
[Variable1] [Trial Start]
[Variable2] [Trial End]
%{Variable1}
%{Variable2}
# This will result in the following data
[Trial Start]
[Trial End]
# You can also use global variables in global variables like this:
[Variable0] Trial
[Variable1] [%{Variable0} Start]
[Variable2] [%{Variable0} End]
%{Variable1}
%{Variable2}
# This will result in the same output as the pervious example. Global variables can also
# be changed in the middle of a file, however all replacements that have happened before
# the change in a global variables value will still use the previous value.

# All commands to the player (except for the Experiment File tag) are given with in
# a trial block
# This is the start tag for a trial
[Trial Start]
# This is the end tag for the trial
[Trial End]
# There are four types of trials fragment, initialize, randomize, and finalize
# fragment trials are not run but provide an away to include common functionality
# in a single place where they can be inserted in any of the trial types including
```

```
# other fragments
# fragment trials have the tag
[Fragment] Param
# Where param can be any string value that will be used as a reference to this
# specific fragment. More on fragments will be given later on.
# initialize trials are all run in order they exist in the file
# before any randomize or finalize trials are run even if the trial comes
# before the initialize trial in the file
# initialize trials have the tag
[Initialize]
# finalize trials are all run in order they exist in the file
# after all the initialize and randomize trials have run even if they
# come after the finalize trial in the file
# finalize trials have the tag
[Finalize]
# If a trial block contains multiple type tags such as fragment, initialize or finalize
# tags the block will be the type of the last tag found in the block so
[Trial Start]
[Finalize]
[Initialize]
[Trial End]
# will be an initialize trial
# If a trial does not contain either a fragment, initialize or finalize tag it is
# a randomize trial. All randomize trials are run after all initialize trials
# and before any finalize trials are run. The order of the trials does not matter
# The order of a randomize trial will be run in a random order with respect to all
# the other randomize trials. Because of this a randomize trial should end in the
# same state that it starts in to prevent other trials from working incorrectly
# since they could be run before or after another given trial
# As stated above fragments can be used in any type of trial. However, it must be
# defined before it can be used, so it must be in the file before the trial that
# is trying to use it. To use a fragment just include the tag
[Add Fragment] Param
# Where the parameter is the same parameter that was given when the fragment was
# created. This tag will place at this point in the trial all tags from the
# fragment trial excluding all tags mentioned in this block, that includes the
# start and end tags, any fragment, initialize, or finalize tags along with the
# add fragment tag which at this point would have already been replaced with that
# fragments trial data.
# An example of something you could do with fragments is if you have a set
# of instructions
# that is shown in a message box (More on message boxes later) that never changes you
# could create a fragment trial for showing the message box such as something like this
[Trial Start]
[Fragment] Instructions
[MessageBox Start]
# Add message box data here
[MessageBox End]
[Trial End]
# Then in your trial all you would have to do to show the message box is add the tag
[Add Fragment] Instructions
# There is another way to add fragments that can be even more useful. You can use
# the add fragment block with start and end tags
[Add Fragment Start] Param
[Add Fragment End]
# Where the parameter is the same as for the normal add fragment tag
# Inside the add fragment block you can define variables with the following tags
[Variable1] Param
[Variable2] Param
[Variable3] Param
[Variable4] Param
[Variable5] Param
[Variable6] Param
[Variable7] Param
[Variable8] Param
[Variable9] Param
[Variable0] Param
# Where the Param is what the variables value will be. In fragments at any call
# level you can use a variable and that variable will be replaced with the parameter
# passed in. However, variables can only be used on normal lines and not special
```

```
# lines like the tags listed in this section. For example you can not use a variable
# to add another fragment. This is because the add fragment tag would have already
# been resolved by the time the variables are replaced with their values.
# Now for the ten different variables they can be used in a fragment with the
# following strings ${Variable1} ${Variable2} ${Variable3} ${Variable4} ${Variable5}
# ${Variable6} ${Variable7} ${Variable8} ${Variable9} ${Variable0}
# So for example if you have the following trials:
[Trial Start]
[Fragment] Hello
[Comment] ${Variable1}
[Trial End]
[Trial Start]
[Add Fragment Start] Hello
[Variable1] How Are you Doing?
[Add Fragment End]
[Trial End]
# This will result in the following trial (The comment tag is explained in the next
section)
[Trial Start]
[Comment] How Are you Doing?
[Trial End]
# Another example of how fragment variables work is:
[Trial Start]
[Fragment] Test 1
[Comment] ${Variable2}
[Trial End]
[Trial Start]
[Fragment] Test 2
[Comment] ${Variable3}
[Add Fragment] Test 1
[Trial End]
[Trial Start]
[Add Fragment Start] Test 2
[Variable2] How Are you Doing?
[Variable3] Hello
[Add Fragment End]
[Trial End]
# This will result in the following trial
[Trial Start]
[Comment] Hello
[Comment] How Are you Doing?
[Trial End]
# This is because when fragment Test 2 is parsed it will resolve to the trial:
[Trial Start]
[Fragment] Test 2
[Comment] ${Variable3}
[Comment] ${Variable2}
[Trial End]
# There are ways that you can group random trials together so they are shown as a group
# instead of mixed in with all the other random trials. To do this you use the following
# tags to define a block of trials to be grouped together.
[Random Start] Param
[Random End]
# These tags need to be placed outside of the trial blocks since they should surround a
# block of trials. The parameter is the id of the block which must be a positive integer
# You can use the same random group id multiple times and all the trials included in
# one of the blocks will be put into a single group under the same id.
# When trials are being shown trials that are not part of a random group are shown first
# then the random group with id 1 will be shown followed by 2, 3, etc. However, you must
# have groups with ids starting at 1 and going to the highest group number without
# missing any integers in between. If you have an empty group it will not be saved. So
# if you have random groups for 1, 2, and 4 only groups 1 and 2 will be shown. All the
# trials in group 4 will not be shown at all. So to get them to be shown you need to
# change the id for group 4 to 3 since that is the next integer expected for the group.

# Some useful tags for writing data to the user movie file that is being recorded
# are the comment tag with parameter and the blank line tag
# The comment tag
[Comment] My comment
# will write the following to the user movie file:
```

```
# My comment
# To add a blank line to the user movie file use the tag
[Blank Line]
# Also you can get the same result if you use the comment tag without a parameter
# that means the new line comes right after the ]
[Comment]

# To make the user movie file more readable the experiment player will add the following
# tags to experiment movie file whenever it comes across the trial end tag to end
# a trial block
[Comment] End of Trial
[Blank Line]

# The following tags are for selecting objects, but first this tag is to deselect all
# objects that are currently selected
[Deselect All]

# You can show a message box to the user by using the message box block
# The blocks start and end tags are
[MessageBox Start]
[MessageBox End]
# The block must contain the following tags
[Title] Param
[Message Line] Param
# The title tags parameter is the title for the message box. If there are
# more then one title tag then the last tag in the block will be used
# as the title for the message box to display
# The message line tags parameter specifies one line of text in the message
# part of the message box. If you have more than one message line tag
# the message box will have multiple lines of text
# There is also a special tag that can be used to show the progress of the
# experiment so the user will know how much they have left to do
[Progress Line]
# This tag will add a line in the format of (Progress: #/# = #%)
# to the list of lines in the same position as if the line had been a message line
# tag instead. For this to work the total number of questions must be already
# defined and question numbers must also be used for the progress to be accurate
# more about question number are later in this document

# There are two ways to set the camera
[Camera] Param
# Will set the camera to the exact position and orientation given in the parameter
# this also allows for free movement controls
# The parameter needs to be in the format of x,y,z,azimuth,elevation
# Where all values x, y, z, azimuth, and elevation are integers or floating point
# values
# The other way to set the camera is using the tag
[Camera Origin] Param
# This will set the camera looking at the origin at the right distance, elevation,
# and rotation angles given in the parameters. This also allows for rotation movement
# controls so you can rotate or tilt around the origin instead of just rotating
# the camera itself like in free movement. Zooming is also allowed, but panning is not
# The parameter needs to be in the format of azimuth,elevation,distance
# Where all values azimuth, elevation, and distance are integers or floating point
# values
# Since it is hard to know these camera values to create the tag for there is an
# easy way to set the camera to the exact view you want. If you are running in
# experiment mode but not test mode you can setup the camera they way you want
# it to be positioned at and then with nothing selected use the right click menu
# and select item "Get Camera Data" this will display a dialog with the whole
# tag that can be copied and pasted into your experiment file. If you want to get the
# camera data from the origin view first you most change to the camera to be in origin
# mode by right clicking with nothing selected and choosing "Set Camera To Origin Mode"
# If you want to change back do the same thing but select "Set Camera To Eye Mode"
# When changing from eye to origin mode the camera is most likely going to change the
# look of the current view since it will now be targeting the origin. Changing from
# origin to eye mode normally causes little change in the view.

# You can show the axis with the tag
[Axis Show]
```

```
# And hide the axis with the tag
[Axis Hide]

# For the following layer visibility commands the parameter is the
# localname of the layer in the owl file. To find an easy way to get the localname
# of a given layer see the information about [Select Node] and [Select Layer]
# You can make a layer visible with the tag
[Layer Visible] Param
# You can make a layer hidden with the tag
[Layer Hidden] Param
# You can make the layers element (The box shape that is the selectable part of
# the layer) visible if the layer is visible with the tag
[Layer Show Element] Param
# You can make the layers element hidden if the layer is visible with the tag
[Layer Hide Element] Param
# Note the state of the element is not kept if you make the layer itself visible
# or hidden after a call to show or hide the element
# So if you have the tags
[Layer Hide Element] Param
[Layer Visible] Param
# The result will be that the layer element is visible. For the tags
[Layer Visible] Param
[Layer Show Element] Param
[Layer Hidden] Param
# The result will be that the layer element is invisible along with the layer

# There are two ways to move a layer. The first way is with the layer move block
# The blocks start and end tags are
[Layer Move Start] Param
[Layer Move End]
# For the start tag the parameter is the localname of the layer in the owl file
# that is going to be moved
# The following tags are optional in the layer move block
[X] Param
[Y] Param
[Z] Param
# The parameters for these tags needs to be an integer or a floating point value
# If the same tag is in the block more then once the last tag in the block will
# be used
# These tags represent the x, y, and z locations to move the layer to
# If a tag is not provided in the block that coordinate will be defaulted to 0
# For example the layer move block
[Layer Move Start] Param
[Z] 15
[X] 3
[Layer Move End]
# Will move the layer to location 3, 0, 15
# The other way to move a layer is by using the tag
[Layer Move] Param
# The parameter is the localname of the layer in the owl file
# This moves the layer to the location 0, 0, 0 and is equivalent to the tags
[Layer Move Start] Param
[Layer Move End]
# and
[Layer Move Start] Param
[X] 0
[Y] 0
[Z] 0
[Layer Move End]

# There are two important types of tags for controlling what the user can do with in
# any given scene the tags
[Enable Controls]
[Enable Input]
# will enable controls and input respectively while the tags
[Disable Controls]
[Disable Input]
# will disable them
# Controls corresponds to the input that allow movement around the scene either with
# the navigation buttons or an input device like a game pad.
```

```
# Input corresponds to the mouse. When input is enabled nodes can be selected. If
# both input and controls are enabled then you can also move around the scene using
# the mouse. So it can be thought that controls work with moving around a scene while
# input corresponds to selecting objects

# At any time during the experiment you can change the background color by using the
# tag
[Background Color] Param
# Where param is the name of the color (Only a few colors are supported) or
# a r,g,b or r,g,b,a where r = red, g = green, b = blue, and a = alpha
# all four values need to integers between 0 and 255

# If you want to pause the execution of the experiment movie file you can use the tag
[Pause] Param
# Where the parameter is an integer value specifying the amount of time to pause in
# milliseconds
# When this tag is encountered no more tags will be handled in the file until the given
# amount of time has passed. This can be useful at the end of a selection task to
# give the user a view of their final selection(s) before it moves on to the next
# trial. If you do not pause then the next trial could start immediately after they
# have made their final selection. However, before you use the pause tag you probably
# want to disable input and controls so they can not move or select something else.
# If they do select something else during the pause time it will not change there
# selection answer because that is recorded as soon as they select the last object
# that meets the selection requirement

# There is a way to provide message box with all the answers and correctness that the
# user got throughout the experiment. A few tags are needed to achieve this
[Question Count] Param
# This tag tells how many questions are in the experiment file. The parameter needs to
# be an integer where it specifies the number of questions. This tag can be included
# in any of the trial blocks, however if it is included more then once the last time
# it is encountered will be the value that it uses for displaying the final results
# When you have a question you want to include in final results you can give it a
# number with one of the following tags
[Question] Param
# Where param is an integer specifying the number of the question
[Question Range] Param
# Where param is in the format of val-val2 where val and val2 are integers where
# val <= val2. This will be the same as calling
[Question] val
[Question] val + 1
...
[Question] val + n
# Where val + n is equal to val2. If val = val2 then it is the same as calling
[Question] val
# The values for the question tags needs to be zero based or one based but should
# not be greater then the value specified in the question count tag
# Using the question tag will store that value or range of values for the question
# range tag into a queue. When a question is encountered in the experiment file
# the first item in the queue will be removed and used as the number for that
# question. So if you can have a set order for all the questions even if the
# questions gone through in a random order. If you use the same question number
# more then once the later use of the number will overwrite the past use.
# There are three type of questions that are recognized. The first is a message
# box with choices, but no answer given. This can be used to collect data such
# as the gender or age of the user. The second is a message box with choices, but
# has a specific answer. The last is a selection block with answers for what
# nodes to select. For the question to be recorded you must first make sure there
# is the right number of question numbers inside the queue before the question
# is asked. If you only have one question number in the queue and then ask for
# the selection of two nodes giving each an answer value only the first answered
# value will be saved as a question to be displayed with the final results.
# There is also ways to add specific column data to an Excel merge of the experiments
# outputted files. The first thing to do is use the question data block instead
# of the question count tag. The question data block has the start and end tags of:
[Question Data Start]
[Question Data End]
# In the block you can use the tag
[Count] Param
```

```
# to set the question count like is done in with the question count tag. Like the
# question count tag the parameter has to be an integer where it specifies the
# number of questions. This should be the number of questions asked to the user
# that results should be recorded for.
# To add a custom or extra column to appear in a created excel file use the tag:
[Extra Column] Param
# Where the parameter is the name of the header for the column. The header can not
# contain a comma or tab inside it, but spaces are allowed however you should avoid
# using more then one space to separate things. Also you can not use the name of
# Default or Number since that is used internally. You can however use the name
# default with a lowercase 'd' or number with a lowercase 'n'. The order that the
# extra column tags are encountered in the question data block defines the order
# they will appear in the resulting excel file. Extra columns will appear between
# the Question and Correct Ans columns. If you include extra columns then every
# question must have data for every single column that have been defined. To do
# this you need to use the question block which has start and end tags of
[Question Start]
[Question End]
# To set what the question number is use the tag:
[Number] Param
# Where the parameter is an integer specifying the number of the question. Like the
# question tag the parameter should be a value in the range of [0, question count]
# If it falls out of that range it will not be included in the merged Excel data
# Since every extra column must be included a quick way to insure this is to use
# the tag:
[Default] Param
# Where the parameter will be the value given to each of the extra columns. Note:
# to use this you must have already defined the extra columns in the question data
# block else no extra columns will be populated with the given value.
# For setting the value of a specific extra column use the tag
[<Column Header>] Param
# Where <Column Header> is replaced with the Param from the extra column tag defined
# in the question data block. The parameter will be the value to set for that column
# for the given question. Now you can include any extra column header data this way
# but if the number of extra columns does not match the columns that were defined
# in the question data block or not every extra column has a value then the Excel
# data will not be stored inside the outputted user movie file.

# By default for an experiment the light source is of type directional (Like the Sun)
# and has a direction angle of (500, 500, 500) or (1, 1, 1) the light has black or no
# ambient but does have diffuse and specular color which is set to white. There is
# also no attenuation on the light. The light can be changed in any way using the
# light block with start end tags:
[Light Start] Param
[Light End]
# The parameter on the start tag is the index of the light. By default there will
# always be one light that is created (Index 0) without any light tags. This light
# can be changed by using the start tag
[Light Start] 0
# If you want to have more then one light you must include increase the index by
# one for every light you want created. A light will only be created if the light
# at index - 1 already exists. So to create three lights you can use the following
# tags:
[Light Start] 0
[Light End]
[Light Start] 1
[Light End]
[Light Start] 2
[Light End]
# Since light index 0 already exists that block could be left out. However, you probably
# want to change the characteristics for the light instead of using the default.
# You will get an error if you try to create three lights this way:
[Light Start] 0
[Light End]
[Light Start] 2
[Light End]
[Light Start] 1
[Light End]
# This is because when it tries to create light index 2 it checks to see if light index
# 1 exists first which it does not since the creation is done after the light 2 block.
```

```
# One more note about light indexes, there can be at least eight lights, and there is
# a chance that more lights could be supported, however you should stick to a max of just
# eight lights (Indexes 0-7) to avoid a problem on a system with a max of only eight
# lights.
# All tags inside the light block are optional. If they are not supplied then the
# current value of the light will be kept. Also if a tag is repeated or contradicts
# another tag the tag closer to the end of the block will be used over a tag earlier
# in the block
# The three colors of the light ambient, diffuse, and specular can be set with the
# following tags inside the light block
[Ambient] Param
[Diffuse] Param
[Specular] Param
# Where the parameter is the color, which can be either a supported color name or
# an RGB or RGBA value in the format of r,g,b or r,g,b,a where the values are
# integers between 0-255
# The direction of a directional light and the position of a positional light or
# spotlight use the same variable so if you set the directional value of the light
# and then change it to a positional light the direction value becomes the position
# value for the positional light and vise versa
# To set the direction of a directional light use the tag:
[Direction] Param
# Where the parameter is a 3D vector in the form x,y,z. So for the default light
# direction you can use the following tag
[Direction] 500,500,500
# Using the direction tag will set the direction of the light, but also set the
# values so that the light set to be a directional light
# To set the position of a positional light or spotlight use the tag:
[Position] Param
# Where the parameter is a 3D vector in the form x,y,z. Using this tag will make sure
# that the light is currently either a positional light or a spotlight. If it was
# previously a directional light it will change to a positional light. However, if it
# was already a positional or spotlight it will keep its current type
# To specifically change the type of the light you can use one of the following tags:
[Directional]
[Positional]
[Spotlight]
# By default the light is always on, but you can turn the light off if you want to.
# However, turning a light off might have bad visual results. The tag
[On]
# Will make sure the light is on while the tag
[Off]
# Will turn off the light
# A light has three different attenuation values, constant, linear, and quadratic.
# These values work together to define how the intensity of the light can fall off
# the farther away you get from the light source. Since directional light has no
# position it always has no attenuation even if these values are set. To have no
# attenuation constant should be one and linear and quadratic both need to be set
# to zero. No attenuation means that the lights intensity is the same no matter
# what the distance is away from the light source is. The three attenuation values
# can be set using the following tags:
[Constant Attenuation] Param
[Linear Attenuation] Param
[Quadratic Attenuation] Param
# Where the parameter is a decimal value defining the value for that attenuation
# component
# The next three tags are for setting up spotlight components. Using any of these
# three tags will change the light type to spotlight
[Spotlight Direction] Param
# This tag will set the direction where the spotlight is facing. The parameter needs
# to be a 3D vector in the form x,y,z
[Spotlight Angle] Param
# This tag will set the cutoff angle for the spotlight. This is the angle from the
# direction where the light will go out to which describes the cone of light for the
# spotlight. The full angle of the cone will be twice the value given because it is
# used once for each half of the cone. The parameter needs to be the angle in degrees
# and should be between 0-90, however, the value of 180 is also valid and will cause
# the same result as the positional light which produces light in all directions
# from its given position.
[Spotlight Exponent] Param
```

```
# This tag sets the exponent for the spotlight. The parameter needs to be between 0
# and 128. This value defines how the lights intensity fades off the farther away
# it gets from the center of the spotlight cone. A value of zero corresponds as
# uniform intensity over the whole cone.
```

Experiment Movie Format (Phase One):

The following describes new additions to the September 2008 experiment movie format from the format listed above for use in the current experiment.

```
-----------------------------------
# Everything from the September 2008 Experiment Movie Format still applies

# The floor/layer in phase one of the experiment has three different images
# so to set the image on a particular layer the following tags can be used:
[Layer Floor Grid] Param
[Layer Floor Lanes] Param
[Layer Floor Ticks] Param
# These tags set the floor image to be a grid, lanes, and tick marks
# respectively. The parameter for each item needs to be the URI of the layer.
# To have the sides/wings and part of the floor of a given lane be highlighted
# you can add a lane letter (A-G) to have the lane highlighted. Note that only
# one lane can be highlighted at a time. The form for the floor tags with a
# highlighted lane is as follows:
[Layer Floor X Grid] Param
[Layer Floor X Lanes] Param
[Layer Floor X Ticks] Param
# Where X is one of the letters: A, B, C, D, E, F, G, or H which tells what
# lane will be highlighted. For example to highlight lane C with a lanes
# marked on the floor use the tag:
[Layer Floor C Lanes] Param

# Turning on and off drop lines can be done with these two tags:
[Drop Lines Show]
[Drop Lines Hide]

# Turning on and off shadows can be done with these two tags:
[Shadows Show]
[Shadows Hide]
# However, since the implementation of shadows is not a one size fits all
# implementation the properties for the shadows can be initialized or changed
# by using the shadows block with start end tags:
[Shadows Start] Param
[Shadows End]
# All tags inside the shadows block are optional. If they are not supplied then the
# current value of the shadows will be kept. Also if a tag is repeated or contradicts
# another tag the tag closer to the end of the block will be used over a tag earlier
# in the block
# Turning on and off shadows can be done in the block by using the tags inside the
# shadows block:
[On]
[Off]
# There are three points/vectors that are used to define where the light source
# is coming from which will cast the shadows. These points/vectors can be set
# with the tags:
[Light Position] Param
[Light Target] Param
[Up Vector] Param
# Where the parameter is a 3D vector in the form x,y,z. The light position defines
# the place where the light is located in 3D space which is needed, because in the
# case of directional lights, which just define the light vector, no position
# is defined for them. The light target is used to tell where the light is pointed.
# Directional, and spot lights are the only lights you should use for casting
# shadows since they have a specific direction the light is going. Positional lights
# are like a lamp and thus shine light in all directions so a single direction can
# not be used to create its realistic shadows. However, this type of shadows could
# be used on a positional light if you only care about shadowing one general
# direction where the light is coming from. The direction of the light for casting
# shadows is defined by the vector formed by going from the light position to the
```

```
# target. The up vector is never needed and in fact if an invalid vector is provided
# (one that is parallel to the direction of the light) a valid one will be computed
# and used in its place. The ability to set an up vector is just in place to help
# frame the created image of the scene from the light sources point of view in
# which case the up vector defines the roll of the camera positioned at the
# light source.
# In the creation of shadows there are a few variables that can tweaked to help
# improve the way the shadows turn out based on the current scene. The first
# two points can be set with the following tags:
[R Offset] Param
[R Scale] Param
# Where the parameter is a floating point value which is defaulted to 0.5. These
# two values are used in the process where the shadow is compared to the object
# currently being rendered to see if it should have a shadow or not. The next
# two variables are used in the creation of the shadows which are then used in
# the comparison just listed. The two variables can be set with the tags:
[Polygon Offset Factor] Param
[Polygon Offset Units] Param
# Where the parameter is a floating point value. These variables can also have
# there usage be turned on or off using the following tags:
[Polygon Offset On]
[Polygon Offset Off]
# The last things about the shadows that can be set using the shadows block is the
# projection to be used in creating the image of the scene from the light sources
# point of view which is then used to define where shadows will be cast. There
# are two types of projections that can be used: orthogonal and perspective. To
# use an orthogonal projection use the tag:
[Ortho] Param
# Where the parameter is a floating point value which represents a zoom level
# of the image as seen from the light source. A value of zero is no zoom, greater
# then zero means the objects will look bigger and less then zero means they
# will be farther away. This value should be greater then zero and can not be equal
# to zero. The orthogonal projection is useful for directional lights since it does
# not create an image with vanishing points meaning that all points of light cast
# in the direction from the viewing plane will be parallel to each other just like
# the definition of a directional light. A perspective projection on the other hand
# is better used for spot and positional lights since by definition all the light
# comes from a single position and not direction. So since the perspective projection
# will create an image with vanishing points (just like normal vision and 3D graphics)
# having the light rays adjust this way is more natural. To use a perspective
# projection use the tag:
[Perspective] Param
# Where the parameter is a floating point value which represents the field of view
# to use in degrees.
# For both types of projections you can adjust them more by using the following tags:
[Width] Param
[Height] Param
[Near] Param
[Far] Param
# Where the parameters are all floating point values. The width and height are used
# to define the view port of the projection and for the best results these values
# should be equal to each other. The near and far values also go together and are
# used to define the near and far clipping planes respectively. These are some of
# the most impotent variables for shadow creation because they help to distinguish
# how each object in the image (based on distance from the light) is likely to be
# a shadow or not.

# A Theory of Signal Detection (TSD) question is a yes/no, true/false question that
# also has a certainty level along with it such as being pretty sure the answer is
# yes. These questions can be asked at any time in the experiment by using the
# TSD question block with start end tags:
[TSD Question Start]
[TSD Question End]
# If any child/sub tag is repeated or contradicts another tag the tag closer to
# the end of the block will be used over a tag earlier in the block. The following
# tag is required in the TSD question block:
[Answer] Param
# Where the parameter is the correct answer to the TSD question which can have
# values where case is ignored of true,yes,1 for a yes answer and false,no,0 for
# a no answer. The other tags in this block are optional.
```

```
# To connect a question with the answer you can use the tag:
[Question] Param
# Where the parameter is string of the question which can be displayed to the
# subject during the experiment when this question is asked.
# Normally when a TSD question is asked it will wait for a TSD response to come
# from the subject before it continues on with the experiment. To limit the
# amount of time the user has to make their selection(s) you can use the tag
[Timeout] Param
# Where the parameter is the amount of seconds which must be a number to give the
# subject to make there choose. If this amount of time passes they will get a time out
# response for the TSD question.
# There is also a special timeout block that can be used with start and end tags:
[Timeout Start] Param
[Timeout End]
# Where the parameter for the start tag works the same as the normal timeout tag.
# What is unique about this timeout block is that almost any tags can go into this
# block and if a timeout happens they will be executed. However, it should be
# noted that what you try to do in this block should be limited and doing things
# like asking another question must be avoided. Also it is possible that the full
# block will not be executed on a timeout which must be kept in mind when using
# this special block.
# Another tag you can use is this:
[Pause] Param
# Where the parameter is the amount of milliseconds which the movie player should
# be paused for after a response has been made.
# A special block for the TSD question is like the timeout block it has the
# start and end tags:
[Response Ready Start]
[Response Ready End]
# This block lists tags that will be performed when the subject has stated to
# the program that they are ready to answer the question. These tags will not
# run if the subject times out. This block shares the same limitations as the timeout
# block so avoid doing things like asking other questions.
# When a timeout happens you can force the subject to give there answer which
# signals the code that the subject said they are ready to answer by including the
# following tag in the TSD question block:
[Force Answer]

# The next two tags where created specifically for showing which node is the target
# node in the scene by changing its color. The two tags are:
[Node Color Red] Param
[Node Color Green] Param
# Where the parameter is the URI of the node to change the color of. The two tags
# will change the color of the node to red and green respectively. So before the
# question is asked you can change the target node to red, and then back to green
# after you are done asking the question.
# The exact RGB color value can be changed for the target node (red) without
# recompiling by using the tag:
[Node Red Color] Param
# Which lets you set the color to be used in the node color red tag. Where the
# parameter is the color, which can be either a supported color name or an RGB
# or RGBA value in the format of r,g,b or r,g,b,a where the values are integers
# between 0-255

# The next tag is used when being viewed on a true 3D display, but has no effect
# on anything when being used on a normal one. To change the eye separation value
# which is the distance between the left eye image and right eye image use the tag:
[Eye Separation] Param
# Where the parameter is a floating point number for the separation value between
# the two eye images.

# To increase the amount of available variables supported from 10 to 16 the following
# variable tags are now supported in the global and fragment space:
[VariableA] Param
[VariableB] Param
[VariableC] Param
[VariableD] Param
[VariableE] Param
[VariableF] Param
# These variables work just the same as the 0-9 variables and are supported as
```

```
# global and fragment variables. These A-F variables can be referenced in the
# same way like the other variables for example using a global A variable use
# %{VariableA} and for a fragment A variable use ${VariableA}
# By using the syntax $EQUALS{Param1~Param2} in a trial this will be replaced
# with true or false based on a string compare what is in parameter 1 and
# parameter 2, which is done at "Runtime" (i.e. after the file has been
# parsed and is being played). It is also useful to know that you can use
# variables as the parameters. Here are two examples of what can be done with
# this type of command:
...
[Answer] $EQUALS{${VariableC}~100}
...
[Answer] $EQUALS{${VariableE}~${VariableF}}
...
# The first example will checks if variable C is equal to 100 and the second
# example checks if variable E and F are equal. If the first example results
# as true and the second as false the two lines will be treated as follows:
...
[Answer] true
...
[Answer] false
...

# To make it so that trials that are similar to each other are not seen back
# to back there is a new tag that can be set for a trial that tells what
# this trial is like. The tag is:
[Random Like] Param
# Where the parameter is a string which is compared to the trial's value.
# When a trial has a random like value set for it the randomness will try
# to position the trial in a random order (only for the block it is in) where
# any trial before or after it (again only in the block and not across blocks)
# will not have the exactly the same random like value (the parameter).

# In the experiment if you don't want the mouse to be shown on the screen you
# can use the tag:
[Mouse Hide]
# To show the mouse again just use the tag:
[Mouse Show]

# By placing the following tag where ever an experiment end tag could go
# (not inside a trial block for example) you can reference another file:
[File End Import] Param
# Where the parameter is the path to the file to import. This import process
# will stop reading the current file (so it is not like programming import or
# include statements) and will continue on to the other file under the under
# standing that a texture replace was done. An imported file can in turn import
# another file and that import can even use global variables set in the
# first import to allow the controlling of multiple imports past the first one.
```